smc 2010

7th Sound and Music Computing Conference

PROCEEDINGS

Emilia Gómez, Perfecto Herrera, Rafael Ramírez (eds)

21-24 July 2010, Barcelona
Universitat Pompeu Fabra

Organizers

Sponsors

Collaborators

smc2010.smcnetwork.org

ISBN: 978-84-88042-72-9

SMC 2010 Committee

General Chair
Xavier Serra (Universitat Pompeu Fabra, Spain)

Scientific Programme Chairs
Emilia Gómez (Universitat Pompeu Fabra and ESMUC, Spain)
Perfecto Herrera (Universitat Pompeu Fabra and ESMUC, Spain)
Rafael Ramírez (Universitat Pompeu Fabra, Spain)

Music Programme Chair
Andrés Lewin-Richter (Phonos Foundation, Spain)

Summer School Chair
Enric Guaus (ESMUC, Spain)

Local Committee
Ferdinand Fuhrmann (Universitat Pompeu Fabra, Spain)
Martín Haro (Universitat Pompeu Fabra, Spain)
Piotr Holonowicz (Universitat Pompeu Fabra, Spain)
Carles F. Julià (Universitat Pompeu Fabra, Spain)
Cyril Laurier (Universitat Pompeu Fabra, Spain)
Gerard Roma (Universitat Pompeu Fabra, Spain)
Alba B. Rosado (Universitat Pompeu Fabra, Spain)
Justin Salamon (Universitat Pompeu Fabra, Spain)
Inês Salselas (Universitat Pompeu Fabra, Spain)
Joan Serrà (Universitat Pompeu Fabra, Spain)
Mohamed Sordo (Universitat Pompeu Fabra, Spain)

Scientific Committee
Josep Lluis Arcos (IIIA-CSIC, Spain)
Federico Avanzini (University of Padova, Italy)
Álvaro Barbosa (Universidade Católica Portuguesa, Portugal)
Dominique Fober (GRAME, France)
Francisco Gómez Martín (Universidad Politécnica de Madrid, Spain)
Fabien Gouyon (INESC Porto, Portugal)
Hendrik Purwins (Universitat Pompeu Fabra, Spain)
Davide Rocchesso (University Iuav of Venice, Italy)
Stefania Serafin (Aalborg University in Copenhagen, Denmark)
Domenico Vicinanza (DANTE and ASTRA/Lost Sounds Orchestra, United Kingdom)
Gualtiero Volpe (InfoMus Lab, Genova, Italy)
Stefan Weinzierl (TU Berlin, Germany)

Reviewers

Christina Anagnostpoulou
Daniel Arfib
Stephan Baumann
Juan Pablo Bello
Emmanouil Benetos
Dmitry Bogdanov
Eoin Brazil
Roberto Bresin
Emilios Cambouropoulous
Antonio Camurri
F. Amílcar Cardoso
Joel Chadabe
Daniela Coimbra
Nick Collins
Sofia Dahl
Roger Dannenberg
Laurent Daudet
Bertrand David
Hugo de Paula
Giovanni de Poli
Simon Dixon
Anibal Ferreira
Paulo Ferreira-Lopes
Arthur Flexer
Ferdinand Fuhrmann
Hiromasa Fujihara
Martin Gasser
Bruno Giordano
Rolf Inge Godoy
Werner Goebl
Maarten Grachten
Holger Grossmann
Enric Guaus
Carlos Guedes
Pierre Hanna
Kjetil Falkenberg Hansen
Martin Haro
Chris Harte
Toni Heittola
Tomas Henriques
Piotr Holonowicz
Andre Holzapfel
Jordi Janer
Sergi Jordà
Martin Kaltenbrunner
Robert Keller
Stefan Kersten
Anssi Klapuri
Olivier Lartillot
Cyril Laurier
Sylvain Legroux
Adam Lindsay
Fernando Lopez-Lezcano
Penousal Machado
Esteban Maestre
Joseph Malloch
Sylvain Marchand
Matija Marolt
Luis Gustavo Martins
Matthias Mauch
Davide Andrea Mauro
Annamaria Mesaros
Tomoyasu Nakano
Bernhard Niedermayer
Katy Noland
Rui Paiva
Bruce Pennycook
Alfonso Pérez
Carlos Pérez-Sancho
Mark Plumbley
Pietro Polotti
Pedro J. Ponce de Leon
Christopher Raphael
Zbigniew Ras
Josh Reiss
Gaël Richard
David Rizo
Fernando Rocha
Martin Rohrmeier
Gerard Roma
Robert Rowe
Justin Salamon
Joan Serrà
Tamara Smyth
Mohamed Sordo
Antonio Sousa-Dias
Koray Tahiroglu
Mari Tervaniemi
Barbara Tillmann
Todor Todoroff
George Tzanetakis
Giovanna Varni
Marcelo Wanderley
Tillman Weyde
Gerhard Widmer
Alicja Wieczorkowska
Kazuyoshi Yoshii

SMC is a privileged forum in Europe for the promotion of international exchanges around Sound and Music Computing.

The SMC initiative is jointly supervised by the following European associations:
• AFIM (Association Française d'Informatique Musicale)
• AIMI (Associazione Italiana di Informatica Musicale)
• DEGEM (Deutsche Gesellschaft für Elektroakustische Musik)
• HACI (Hellenic Association of Music Informatics)
• SITEMU (Sociedad(e) Ibérica de TEcnología MUsical)

Table of Contents

Oral Presentations

Melody and Harmony

Timbre and Melody

Multimodality

Sound Modeling and Processing

Music Classification and Annotation

Interaction

Rhythm and Percussion

Voice and Singing

Poster Presentations

Poster Session 1

Poster Session 2

Poster Session 3

EXPLORING COMMON VARIATIONS IN STATE OF THE ART CHORD RECOGNITION SYSTEMS

Taemin Cho, Ron J. Weiss and Juan P. Bello
Music and Audio Research Laboratory (MARL)
New York University, New York, USA
{tmc323,ronw,jpbello}@nyu.edu

ABSTRACT

Most automatic chord recognition systems follow a standard approach combining chroma feature extraction, filtering and pattern matching. However, despite much research, there is little understanding about the interaction between these different components, and the optimal parameterization of their variables. In this paper we perform a systematic evaluation including the most common variations in the literature. The goal is to gain insight into the potential and limitations of the standard approach, thus contributing to the identification of areas for future development in automatic chord recognition. In our study we find that filtering has a significant impact on performance, with self-transition penalties being the most important parameter; and that the benefits of using complex models are mostly, but not entirely, offset by an appropriate choice of filtering strategies.

1. INTRODUCTION

Chords are defined by the occurrence of harmonically related musical notes, either simultaneously or in quick succession. They are the smallest and most fundamental structures of the tonal system, which, it can be argued, makes them particularly adept at representing western popular music. Therefore, their identification is of great importance to a wide variety of applications in computer music, information retrieval and musicology. Chord transcriptions, however, are not readily available for most recorded music and can only be generated by highly-trained musicians. This motivates the development of automatic approaches to chord recognition, and explains the interest that this task has generated on the music computing community over the last decade.

A number of approaches to automatic chord recognition are discussed in the literature, several of which are mentioned in this paper. Notable amongst those, are the works of Fujishima [1], which introduced the use of chroma features to represent signal content, and of Sheh and Ellis [2], which pioneered the use of hidden Markov models (HMM) for chord recognition. It can be argued that all subsequent works are variations of the *standard* approach defined by both those papers, comprising a combination of chroma feature extraction, filtering and pattern matching, as shown in Figure 1. This is attested to by the homogeneity of most submissions to the yearly MIREX chord recognition task.

A direct consequence of this overlap is the existence of a number of important system variables that cut across approaches. Understanding these variables, their relation and relative importance in recognition results, is necessary in order to assess the limitations of the standard approach. Unfortunately, save a few exceptions (e.g. [3]), the literature lacks a comprehensive and holistic assessment on how parameter changes affect chord recognition.

The goal of this paper is to investigate the effect of common variables and to reveal their interrelationships, thereby providing valuable information that can guide future developments in automatic chord recognition. To this end, we perform a systematic evaluation including the most common and distinguishable variations in the literature. It must be clarified that covering the full range of possible variations is beyond the scope of this study. Instead we choose to use standard techniques for feature extraction and filtering, and concentrate our evaluation on variations of filtering parameters and the testing of different pattern matching approaches.

The remainder of this paper is organized as follows: Section 2 introduces the standard approach and discusses common variations at each stage; Section 3 presents the evaluation methodology and discusses the experimental results; while Section 4 includes our conclusions and directions for future work.

2. ARCHITECTURE OF A CHORD RECOGNITION SYSTEM

Most chord recognition systems share a common architecture comprised of four main stages: feature extraction, pre-filtering, pattern matching and post-filtering as seen in Figure 1. We discuss each of these steps in detail in the following sections.

2.1 Feature Extraction

The most popular features used for chord recognition are 12-dimensional Pitch Class Profile (PCP), or chroma features [1–12]. Chroma features represent the energy present in each of the twelve pitch classes, and are typically derived by

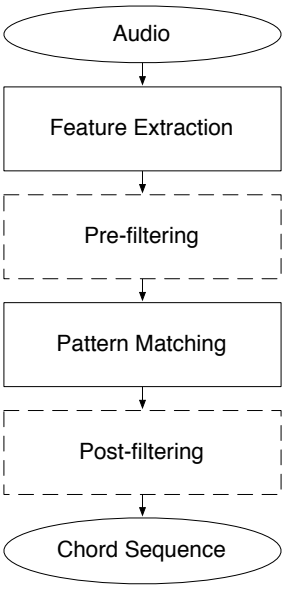
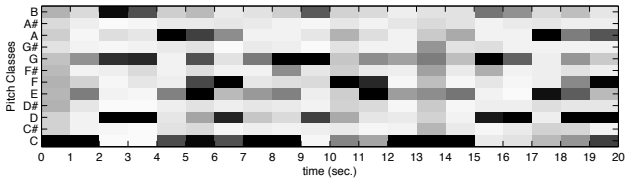
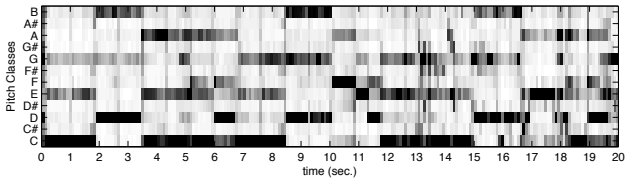


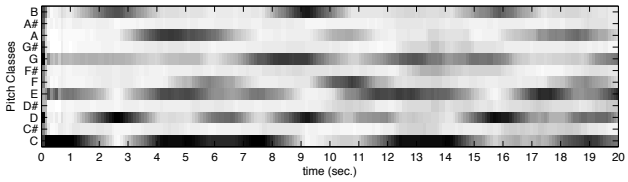
Figure 1. The basic architecture of the standard approach to automatic chord recognition (dotted lines indicate optional steps).



(a) 1 second hop size



(b) 96 ms hop size



(c) 96 ms hop size after smoothing with $N = 15$

Figure 2. Chromagrams computed from the first 20 seconds of "Let It Be".

mapping each frequency bin of the Discrete Fourier Transform (DFT) spectrum to a corresponding pitch class. Many variations on chroma feature extraction are documented in the literature [4, 8, 9, 13, 14].

The most common approach is based on the constant-Q transform, a spectral analysis technique in which the frequency channels are spaced logarithmically [15]. The constant-Q transform X_{cq} of an audio fragment $x(n)$ can be calculated as:

$$X_{cq}(k) = \sum_{n=0}^{L(k)-1} w(n,k)x(n)e^{-j2\pi f_k n} \qquad (1)$$

where k is the bin position, $w(n,k)$ is a window function of length $L(k)$, and f_k is the center frequency of the k^{th} filter bank. The calculation of the center frequency f_k is based on the frequencies of the equal tempered scale with:

$$f_k = 2^{k/\beta} f_{\min} \qquad (2)$$

where β is the number of bins per octave, and $f_{\min}$ is the minimum analysis frequency. From the constant-Q spectrum X_{cq}, the β-bin constant-Q chroma can be calculated as:

$$C_{cq}(b,n) = \sum_{m=0}^{M} |X_{cq}(b+m\beta,n)| \qquad (3)$$

where $b \in [1,\beta]$, and M is the total number of octaves in the constant-Q spectrum determined by the maximum analysis frequency $f_{\max}$. Finally the dimensionality of the β-bin chroma features computed in (3) is reduced to 12 bins by averaging adjacent bins using $\beta/12$-wide non-overlapping Gaussian windows.

The noteworthy thing in this stage is that most systems extract a chromagram with fixed frame rates (i.e. hop size) of 24 - 256 ms, which is significantly faster than the typical rate of chord changes in music. Only a few systems (e.g. [10]) use chroma features extracted from segmented audio frames with variable (and often slower) rates such as beat-synchronous chroma. Errors in the initial feature analysis, e.g. onset detection or beat tracking, in such systems propagate through the subsequent processing stages and can hurt overall performance.

In this paper we follow the fixed frame rate approach in order to focus on the effects of the remaining processing stages on overall performance. We use β = 36, with the analysis performed between f_{min} = 65.4Hz and f_{max} = 1046.5Hz [1] . The resulting window length and hop size are 8192 (186 ms) and 4096 (93 ms) samples respectively at 44100 Hz sample rate.

2.2 Pre-filtering

In order to precisely identify chord boundaries the frame rate of the chroma features must be faster than the rate of chord changes in a piece of music. This is demonstrated in Figure 2 (a) and (b) which compare chromagrams computed using 1 second and 96 ms hop sizes, respectively. The lower precision in Figure 2(a) is clearly visible. For example, the chord change at 11.5 seconds visible in Figure 2(b) is not present in Figure 2(a) at all. Moreover, the longer window exaggerates the influence of transient noise. For example, the short burst of noise in the A pitch class at 14 seconds in Figure 2(b), is drawn out over a full second in Figure 2(a).

However, the disadvantage of using such a short window is that the frames of the resulting chromagram are independent of the long term trend of the signal and respond to local changes, thus becoming sensitive to transients and noise in the signal. A popular technique to cope with this problem is to pre-process the chromagram prior to pattern matching using a low pass filter [1, 3–5, 8, 9, 11]. In this paper, we use

[1] In popular music, the harmonics of a musical note are usually stronger than the non-harmonic components up to 1 kHz [16].

a moving average filter which can be calculated as follows:

$$\hat{c}(n) = \frac{1}{N} \sum_{\tau=0}^{N-1} c\left(n + \tau - \frac{N-1}{2}\right) \quad (4)$$

where $c(n)$ is a frame of the chromagram, $\hat{c}(n)$ is a frame of the smoothed chromagram and n is the frame index. In this paper, $N = 0$ is defined as no filtering.

Intuitively, this technique improves pattern matching because it minimizes the effect of transients and noise in the signal by smoothing the features across neighboring frames. Figure 2(c) shows an example of the output of this process. The noise between 13 and 15 seconds in Figure 2(b) are filtered out in Figure 2(c).

2.3 Pattern Matching

The function of the pattern matching stage is to measure the fit of a set of predefined chord models, corresponding to each of the 24 major and minor triads, to each frame of the input chromagram, thus classifying each frame as being one of the 24 chords. The two most common approaches are based on a deterministic chord template generated by hand or a probabilistic chord model trained from examples of real music. The former approach is quite simple, but many variations of the probabilistic approach have been proposed. In this paper, we evaluate deterministic chord templates and three probabilistic chord models.

2.3.1 Binary chord template

A binary chord template is the simplest and one of the most popular chord models [1, 3–5, 7, 8]. This deterministic chord model is manually generated based on knowledge of the notes used in musical chords. In a binary chord template vector, each component corresponding to a chord-tone [2] is set to 1, and the other components are set to 0. [3] While some systems use variations of the binary chord template that incorporate information about higher harmonics produced by each chord-tone, recent studies have shown that simple binary chord templates are sufficient to obtain a good level of accuracy [7].

2.3.2 Probabilistic chord models

More sophisticated chord models are created by defining probability distributions for each chord class. A popular choice is the multivariate Gaussian distribution. In some systems, the Gaussian chord models are defined manually as with binary templates [3, 5]. More commonly, the distribution parameters are estimated from labeled data.

More precise chord models in the form of Gaussian mixture models (GMM) are sometimes constructed instead of single Gaussian models [12, 17]. Such models use multiple Gaussian distributions to represent each chord. Different components represent more nuanced instantiations of each chord in the training data, producing a more precise fit. This comes at the cost of requiring more sophisticated training using an Expectation-Maximization (EM) algorithm and increased computation when computing probabilities.

Finally, Khadkevich and Omologo [12] use a more complex chord model based on hidden Markov models (HMM) which enforces temporal continuity constraints not present in the other chord models. This chord model follows the topology typically used in automatic speech recognition, consisting of a 3-state, left-to-right HMM with the emission probability under each state consisting of a GMM.

All of the probabilistic models described in this section use Gaussian distributions with either diagonal or full covariance matrices. In most cases a diagonal covariance matrix is used under the assumption that the feature vector components are uncorrelated. In contrast, full covariance matrices capture correlations between different pitch classes. In this paper, we prepared two versions of probabilistic chord models, each with a diagonal and a full covariance matrices. For GMM chord models, we use 5, 10, 15, 20 and 25 mixture components.

The trained Gaussian and GMM models are denoted $M\ell_{\text{cv}}$, where ℓ specifies the number of components in the model and cv specifies the type of covariance matrix, diag or full; e.g. a single Gaussian model with diagonal covariances is referred to as $M1_{diag}$ and a 5 component GMM with full covariance is $M5_{full}$. The 3-state HMMs are denoted as $H\ell_{\text{cv}}$, and follow the same conventions as GMMs. For HMM models, ℓ indicates the number of mixture components used in each state of the model.

The parameters of the chord models are estimated from annotated training data using the EM algorithm. During training, the training data is segmented into 12 major triad and 12 minor triad segments based on the chord annotations. In order to compensate for the limited amount of training data, every chord segment is transposed to the key of C and this key-normalized data is used to train C-major and C-minor models. The trained chord models are then re-transposed to the remaining 11 major and 11 minor keys to define the remaining chord models.

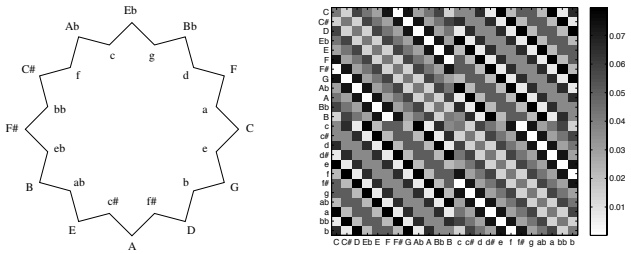
2.4 Post-filtering

The post-filtering stage shown in Figure 1 is used to smooth the sequence of predicted chord labels over time, thereby minimizing the number of spurious chords that only last for a small number of frames. Such mis-detections can be caused by short bursts of noise, which are very common in real music signals (e.g. see in Figure 2(b)). In most systems, post-filtering is performed with a Viterbi decoder, while some systems based on chord templates use a median filter instead [4, 7]. The Viterbi decoder finds the most likely sequence of chords based on the chord-type probabilities computed in the pattern matching stage.

The performance of the Viterbi decoding process is determined by the transition probability matrix, which describes the first-order temporal relationship between chords. Many researchers have concentrated on finding the optimal setting for these parameters. One approach has been to generate the matrix manually based on knowledge of music theory [3], while others have estimated the transition probabilities from music annotations [18].

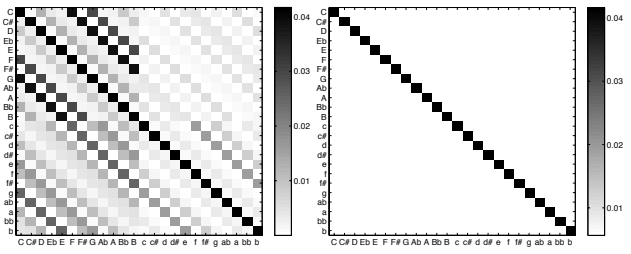
[2] The pitches which make up a chord are called chord-tones and any other pitches are called non-chord-tones.

[3] For example, the binary template for a "C Maj" triad is [1 0 0 0 1 0 0 1 0 0 0 0] where the left to right order of the vector components follows the twelve-tone equally tempered scale from C.



(a) (b)

Figure 3. (a) Doubly-nested circle of fifths, (b) Chord transition probability matrix T_{C5}.



(a) (b)

Figure 4. (a) T_B and (b) T_U, both for 24 chord detection task after applying the transition penalty $P = 2.0$

Much of the previous work contains little discussion of the self-transition probability, which describes the probability of staying in the same chord from frame to frame. While some previous work reports improving the accuracy rate by manipulating the transition matrix, we argue that most of the improvement is contributed by a relatively high self-transition probability, which essentially acts to minimize the number of chord transitions. In a fast-frame-rate analysis, the probability of remaining in a chord is larger than that of moving to another chord, since the rate of chord changes is much slower than the frame rate. Thus, finding the optimal parameter for the self-transition probability tends to have more influence than the other parameters.

To evaluate this assumption, we define the transition penalty P, which is widely used in HMM-based speech recognition systems. This penalty adjusts the strength of the self-transition probability relative to transitions between different chords. It is applied as follows:

$$\log(\hat{a}_{ij}) = \begin{cases} \log(a_{ij}) - \log(P) & \text{for } i \neq j \\ \log(a_{ij}) & \text{for } i = j \end{cases} \quad (5)$$

where $A = [a_{ij}]$ is the original transition probability matrix and $\hat{A} = [\hat{a}_{ij}]$ is the modified matrix with penalty P.

In the case of the 3-state HMM model, this penalty is only applied to the transitions between chords, not transitions within the 3 states comprising the chord model. Since the internal HMM states already enforce temporal continuity within a chord (similar to self-transition probability), we set the diagonal entries of the chord transition matrix used for HMM post-filtering to zero. Therefore, P only changes the transition probabilities between different chord HMMs without touching the internal transitions between the 3 states.

We also evaluate the effect of different transition probability matrices on performance. As a baseline we define a uniform transition matrix T_U in which all transitions have the same probability (1/24 in our task).

For the transition probability matrix derived from music theory, we define a circle of fifths transition matrix T_{C5}, as proposed by [5]. In T_{C5}, the transition probability between two chords is derived from the distance between two chords in the doubly-nested circle of fifths (see Figure 3(a)). In this paper, for a fair evaluation of the parameter P (described in Section 2.4) against T_U, the diagonal entries of T_{C5} (self-transition probabilities) are adjusted to 1/24 and the other members in each row are normalized to sum to 23/24 (see Figure 3(b)).

Finally, we define a transition matrix estimated from training data, T_B. T_B, is estimated from bigrams of symbolic data in the training set. As in the training procedure for the chord models described in Section 2.3, we only calculate chord transitions relative to the current chord, i.e. we assume that all transitions happen from a root of C major or C minor. For example, the transitions C $\rightarrow$ Am and E $\rightarrow$ C#m are both counted as C $\rightarrow$ Am (**`I` $\rightarrow$ `vi`**).[4] The key-normalized bigrams are then transposed to the other major and minor roots to form the final matrix. We adjust the diagonal entries to 1/24 as we do for T_{C5}. Figure 4 shows an example of the transition matrices T_B and T_U after applying the transition penalty P using Eqn. (5).

3. EXPERIMENTS

In this section we describe a series of experiments to evaluate the effect of each processing stage on chord recognition performance. We evaluate the system variations using the well-known Beatles data set, 180 annotated songs from 12 Beatles albums (containing 13 discs).The ground truth chord annotations of the songs are kindly provided by C. Harte [19].[5] The evaluations are performed on 12 major, 12 minor and a no-chord detection task.

Each experiment is performed using a 13-fold cross validation. For each fold, one album is selected as a test set, and the remaining 12 albums are used for training. The chord recognition rate is calculated as follows:

$$\text{Accuracy} = \frac{\text{total duration of correct chords}}{\text{total duration of dataset}} \times 100\% \quad (6)$$

and is averaged across all cross-validation folds.

3.1 Pattern matching without filtering

In order to isolate the power of different pattern matching techniques from the effect of the other processing stages we evaluate the different chord models described in Section 2.3

[4] In this paper, Roman Numerals are open used to indicate the harmonic relationship between two chords without reference to actual chord symbols. In this notation, the first seven Roman Numerals represent a major scale degree from the root. Capital letters are used for major triads, while lowercase letters are used for minor triads, and a flat(b) or sharp(#) in front of a Roman Numeral lowers or raises the diatonic pitch by a half step. e.g. Both C $\rightarrow$ Am and E $\rightarrow$ C#m can be expressed as **`I` $\rightarrow$ `vi`**

[5] http://isophonics.net/content/reference-annotations-beatles

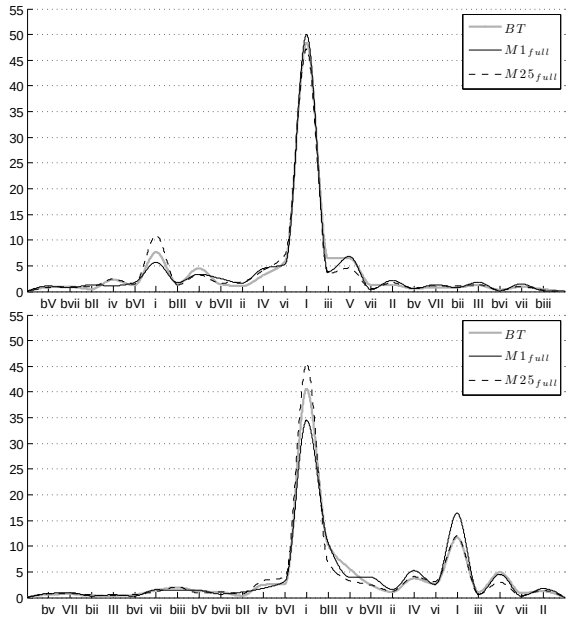


Figure 5. Chord classification errors from the experiment described in Section 3.1. The results are key-normalized to major (top) and minor (bottom) triads and averaged across all roots. The center labels (`I` of major triad detection and `i` of minor triad detection) represent correct classification and the remaining labels represent misclassified chords in Roman Numeral notation. The order of the labels follows the order of the doubly-nested circle of fifths in Figure 3(a).

without performing pre- or post-filtering. The 3-state chord HMM is excluded, because it requires post-filtering. The results are shown in Table 1.

Gauss. #		*1*	*5*	*10*	*15*	*20*	*25*	*BT*
M	*full*	46.8	45.0	44.7	45.1	**46.9**	46.8	46.69
	diag	40.1	42.1	42.6	43.6	45.5	45.5	

Table 1. Average accuracies of pattern matching methods with different chord models without filtering.

It would be reasonable to expect that performance should improve with increasing complexity of the chord model. However, the results in Table 1 contradict this expectation. In fact, none of the trained probabilistic models significantly outperform the simple binary template BT. All of the probabilistic models using diagonal covariance perform worse than BT, while those that utilize full covariance have performance roughly on par with BT. Another surprising result is that the performance of the GMM systems is no better than that of the single Gaussian chord model $M1_{full}$.

All of the chord models based on full covariance matrices perform better than the corresponding models based on diagonal covariance matrices. This gap is reduced by increasing the number of mixture components in the GMM. This trend is almost perfectly maintained across all experiments, therefore we only report results using full covariance matrices in the remaining experiments.

The distribution of chord detection errors is shown in Figure 5. The majority of errors are the result of confusions between harmonically related chords, i.e. those which share the same notes. For example, the chords in a parallel relationship (`I` and `i`), share a common root and fifth, and have only a semi-tone difference between the thirds. The figure also shows that actually $M25_{full}$ outperforms $M1_{full}$ and BT on minor chord detection. However, this is not reflected in the total overall results shown in Table 1, only 25% of the test set is comprised of minor chords, and so the decreased performance on major chord constributes more to the average performance.

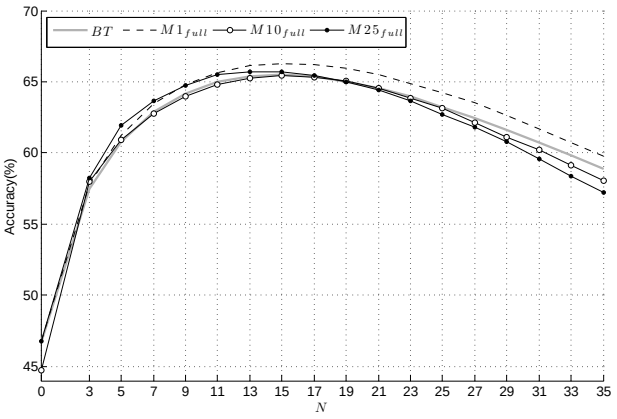


Figure 6. Average chord detection accuracy as a function of moving average pre-filter order N.

3.2 Effect of pre-filtering

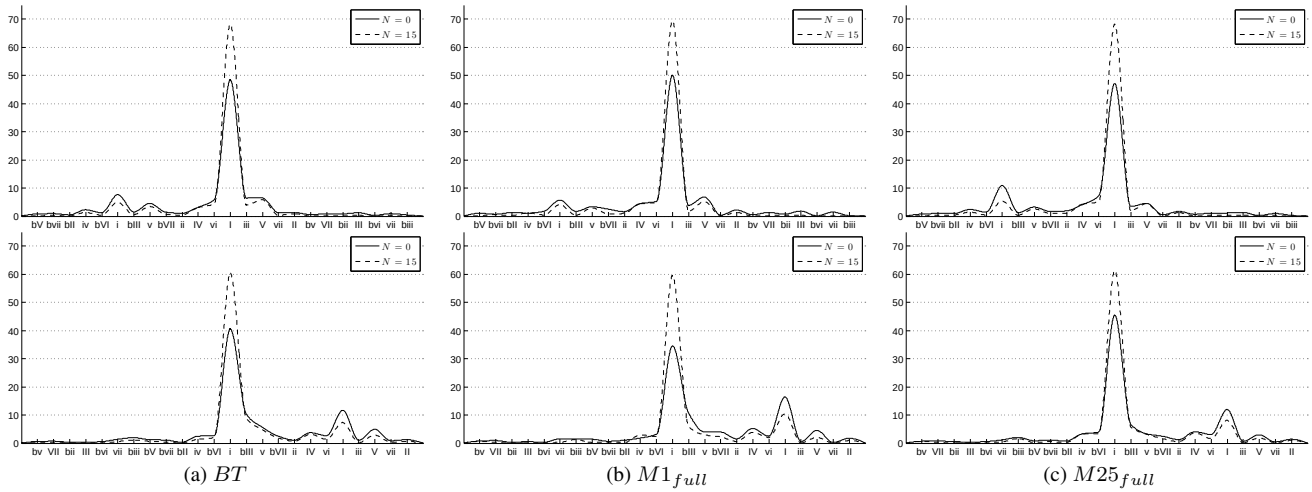
In this section we evaluate the effect of the pre-filtering procedure described in Section 2.2 on chord recognition performance. The chord models are retrained for each setting of the smoothing filter length N, and the filtered chromagrams are used for testing. The 3-state HMM template is excluded for the reasons described in Section 3.1.

The results are shown in Figure 6. Overall, pre-filtering improves performance over the results in Section 3.1 by about 20%. However, the best results for all chord models are almost the same (65.6% $\pm$ 0.1) except $M1_{full}$ (66.3%). The optimal N values for the chord models are also similar ($N = 15$). Once again, the number of mixture components in the GMMs has little effect on performance. The highest accuracy came from the simplest probabilistic model.

Many of the mis-classified frames in the previous experiment consist of very short ($1 - 5$ frame) segments caused by transient noise similar to that shown in Figure 2(b). The large improvement shown in this section is due to the fact that the pre-filtering process largely suppresses this noise.

It seems that training hurts the performance due to overfitting to the small training set and to reduced data variance caused by smoothing the features. In Figure 6, the accuracies of the 25 Gaussian models decrease faster than the single Gaussian models with increasing N. In other words the smoothed data tends to lead to overfitting, especially for chord models containing a large number of parameters.

Figure 7 shows the distribution of chord errors. As expected, a large portion of the performance improvement relative to the results reported in Section 3.1 is the result



(a) BT (b) $M1_{full}$ (c) $M25_{full}$

Figure 7. Major(top) and Minor(bottom) distribution of detections for different N

of fewer confusions between harmonically related chords. As a result, all the distributions of detections are similar to each other at the optimal N. The trained Gaussian model doesn't show any significant improvement against the binary template in this experiment.

3.3 Effect of post-filtering

In this experiment we evaluate the use of the different transition probability matrices defined in Section 2.4 and the effect of the transition penalty P. The pre-filtering parameter N is therefore fixed to 0 while P is varied. In order to smooth the output of BT, pseudo probabilities are calculated by taking the reciprocal of the Euclidean distances between chromagram frames and the chord templates. These are then passed through the Viterbi decoder.

Gauss. #		*1*	*5*	*10*	*15*	*20*	*25*	BT
T_U	P	15	23	22	20	19	15	1.25
	M	67.9	70.3	69.6	69.3	73.2	74.4	70.4
	P	16	18	14	11	8	9	
	H	67.4	71.7	72.5	72.7	74.4	75.0	
T_{C5}	P	15	22	22	20	19	16	1.5
	M	68.0	70.4	69.8	69.6	73.3	74.7	70.2
	P	16	18	14	10	9	8	
	H	67.6	71.9	72.6	72.9	74.5	75.0	
T_B	P	15	22	21	20	18	16	2.5
	M	68.5	70.7	70.3	70.2	73.8	75.1	70.6
	P	16	15	16	10	9	8	
	H	68.2	72.2	73.0	73.5	75.1	**75.6**	

Table 2. Accuracy of each chord model using different chord transition matrices. Each model-transition matrix combination is shown with the P value that maximizes accuracy.

The results are summarized in Table 2. For all systems, post-filtering shows a significant improvement over the results in the previous section, especially in the case of GMM chord models. Another surprising trend is that there is little difference in performance between the different transition matrices. The largest difference between them is only about 0.8%.

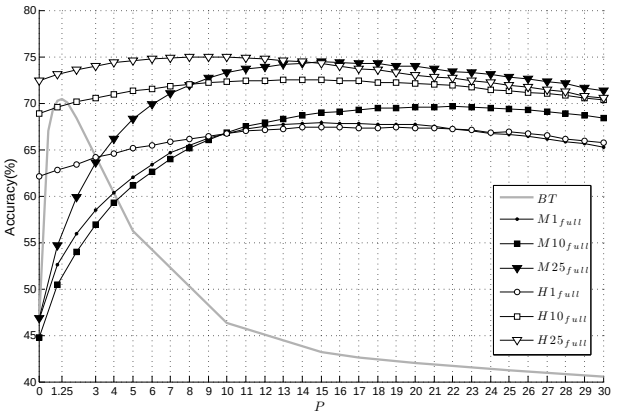


Figure 8. Average accuracy as a function of P using the T_U transition matrix ($N = 0$).

In addition, as shown in Figure 8, the parameter P has different effects on each of the chord models. The BT plot has a very steep curve and peaks at a smaller setting than any other model ($P = 1.25$). The system is overly sensitive to the transition penalty because the likelihoods of each chord class under this model tend to be very close together (i.e. it is not very discriminative), so, for large P, the transition probability overwhelms the likelihood.

On the contrary, the HMM chord model is much less sensitive to P than the other systems. This is due to the smoothing caused by the high internal self-transition probabilities already present within the chord models. The maximum accuracies of $H25_{full}$ and $M25_{full}$ are almost identical at each optimal P value. The two curves match up quite well if that of $M25_{full}$ is moved about 10 units to the left.

3.4 Combined pre- and post-filtering

Finally, we explore the relationship between pre- and post-filtering and evaluate all possible parameter combinations.

G. #	*1*	*5*	*10*	*15*	*20*	*25*	*BT*
N, P	7, 18	3, 23	3, 19	3, 18	3, 21	3, 17	3, 2.5
M	70.4	73.0	74.4	74.7	75.1	75.4	70.8
N, P	9, 7	3, 18	3, 13	3, 11	3, 11	3, 10	
H	70.2	73.7	74.9	74.9	75.4	**75.7**	

Table 3. The best accuracy of each chord model with T_B and optimal N and P.

Since, as described in Section 3.3, the exact transition matrix has a very small effect on performance, we only utilize T_B in this experiment.

The optimal N, P parameter combination for each system is shown in Table 3 along with the corresponding chord recognition accuracy. The best results occur using relatively little pre-filtering ($N = 3$) with transition penalty P similar to the optimal settings found in the previous experiment. The combination of both filtering stages has minimal effect on the best performing system, however it does bring the performance of the systems based on simpler GMM chord models to the same level as the HMM systems. From this we can conclude that the additional pre-filtering stage can be used in place of the more computationally complex HMM chord model without significantly affecting performance.

Compared to the results of post-filtering alone in Table 2, pre-filtering increases the accuracy of all chord models. The smoothing across very short frames ($N = 3$) yields significantly improved accuracy, especially in the case of GMM chord models. However, this effect decreases as number of Gaussians increases. For $M25_{full}$ the smoothing has no significant effect on performance. Overall, the combination of pre- and post-filtering decreases the difference in performance between the GMM chord models. More than doubling the size of the model from 10 to 25 mixture components increases performance by only 1%.

A similar trend can be seen in the performance of the HMM systems. More notably, the small gaps (less than 1%) between GMM and HMM systems with the same number of Gaussians implies that additional pre-filtering can compensate for the additional smoothing present in the more complex model. The 3-state left-to-right HMM architecture requires that each recognized chord have a minimum length of 3 frames. As described in the previous section, this has an implicit smoothing effect and provides better time-persistence than a single frame level chord detection. The results in Table 3 demonstrate that the overall effect is similar to that of the 3 frame moving average filter.

3.5 Summary

Table 4 summarizes the experiments described in this section. All combinations of filtering strategies and pattern matching techniques are shown. T-tests show that differences in accuracy greater than 1% are statistically significant ($p < 0.01$). The performance improvements seen when moving down each column demonstrate the very large impact of filtering on the accuracy of chord recognition system. Post-filtering has the largest impact, primarily due to the self-transition penalties, and the combination of pre- and post-filtering leads to relatively small improvement over the optimal choice of post-filter alone, however this improvement was only statistically significant for $M1$ and $H1$. Despite the great deal of research investigating different chord models, we have observed relatively small performance differences between the models in our experiments. Given optimal filtering settings, all systems perform within 5% of each other (bottom row of Table 4). It is worth noting that the chord models used most often in the literature are based on diagonal covariance matrices [2, 6, 12, 17, 18]. In our experiments, diagonal covariance models performed an average of about 2% lower than the corresponding models based on full covariance matrices.

Chord Models	BT	$M1$	$M25$	3-state HMM	
				$H1$	$H25$
No Filtering	46.69	46.76	46.77		
Pre-filtering	65.50	66.27	65.72		
Post-filtering	70.60	68.52	75.14	68.16	75.56
Pre & Post	70.82	70.44	75.40	70.23	**75.70**

Table 4. The best results of each chord model for each experiment.

There is a general trend of improving performance with increasingly complex chord models, but the effect is dominated by the number of parameters used by each model, i.e. the number of mixture components or states. The best results are obtained using the most complex system based on HMM chord models. The system contains the largest number of parameters and therefore has the highest risk of overfitting to the small training data set used in these experiments.

4. CONCLUSION

This paper presents a systematic evaluation of increasingly complex variations of the standard approach to automatic chord recognition, with a focus on the impact of filtering and pattern matching strategies. Experimental results show that filtering, both before and after pattern matching, has a significant impact on the accuracy of recognition. In the case of pre-filtering we found that variations of the parameter N have a similar effect across different models, with the optimal value increasing performance by as much as 20% upon the unfiltered case. Optimal post-filtering can increase performance by little less than 30%, with high self-transition probabilities (determined by the parameter P) being entirely responsible for this change. Unlike N, optimal values of P have to be carefully chosen for each different model. Combining pre- and post-filtering brings about only marginal improvement over the optimal choice of post-filtering. Surprisingly, we found that the effect of different transition matrices is negligible. This indicates that, at fast frame-rates, any attempts to encode information about likely chord transitions is rendered moot by the need to enforce continuity in the estimations.

While the best overall results are obtained for $M25_{full}$ and $H25_{full}$, it is worth noting that the benefits of using

complex models are mostly offset by an appropriate choice of filtering strategies. Notably, the best performance using simple binary template matching is only 5% less than the best performance using a network of 3-state HMMs, with mixtures of 25 full-covariance Gaussians per state. This is troublesome not only because of the significant difference in computational cost and overall complexity between these two models, but also because the extensive testing and parameter selection on such a small and homogeneous dataset most likely means that the difference is attributable to overfitting and is not generalizable to other music.

Of course, these findings are only valid for fast, fixed-rate features, and could be alleviated by slowing down the feature rate, or by using variable-rate methods such as beat segmentation. However, the former will have an impact on accurate detection of chord transition boundaries (which, if increases in accuracy are forthcoming, might be less of an issue), while in the case of the latter, it is far from clear that the current state of the art in beat tracking can ensure robust performance across a wide variety of music, and thus avoid issues of error propagation.

In either case, it is worth pointing out that lower feature frame-rates will considerably reduce the amount of available data, which will in turn negatively affect our ability to train complex models, such as the ones considered in this paper. This, together with the above mentioned issues of overfitting (which we believe to be widespread in chord recognition research), highlights the need for data collection as a necessary step in the development of better approaches. More data will also support the use of discriminative models for pattern matching (which has proven successful in MIREX-09 [20]) and for supervised training of complex dynamic models integrating rhythmic, harmonic and structural analysis [10]. Finally, we have yet to evaluate the impact of different feature extraction strategies and test new methods (e.g. [14]) that have already shown promise in related music analysis tasks.

5. ACKNOWLEDGMENTS

This material is based upon work supported by NSF grant IIS-0844654 and by IMLS grant LG-06-08-0073-08.

6. REFERENCES

[1] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proc. ICMC*, pp. 464–467, 1999.

[2] A. Sheh and D. Ellis, "Chord segmentation and recognition using em-trained hidden markov models," in *Proc. ISMIR*, pp. 185–191, 2003.

[3] H. Papadopoulos and G. Peeters, "Large-scale study of chord estimation algorithms based on chroma representation and hmm," in *Proc. CBMI*, pp. 53–60, 2007.

[4] C. Harte and M. Sandler, "Automatic chord identification using a quantised chromagram," in *Proc. of the Audio Engineering Society (AES)*, 2005.

[5] J. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *Proc. ISMIR*, pp. 304–311, 2005.

[6] M. Ryynänen and A. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, vol. 32, no. 3, 2008.

[7] L. Oudre, Y. Grenier, and C. Févotte, "Template-based chord recognition: influence of the chord types," in *Proc. ISMIR*, pp. 153–158, 2009.

[8] K. Lee, "Automatic chord recognition using enhanced pitch class profile," in *Proc. ICMC*, 2006.

[9] M. Khadkevich and M. Omologo, "Phase-change based tuning for automatic chord recognition," in *Proc. of Digital Audio Effects Conference (DAFx)*, 2009.

[10] M. Mauch, K. C. Noland, and S. Dixon, "Using musical structure to enhance automatic chord transcription," in *Proc. ISMIR*, pp. 231–236, 2009.

[11] H. Papadopoulos and G. Peeters, "Simultaneous estimation of chord progression and downbeats from an audio file," in *Proc. ICASSP*, pp. 121–124, 2008.

[12] M. Khadkevich and M. Omologo, "Use of hidden markov models and factored language models for automatic chord recognition," in *Proc. ISMIR*, 2009.

[13] M. Stein, B. M. Schubert, M. Gruhne, G. Gatzsche, and M. Mehnert, "Evaluation and comparison of audio chroma feature extraction methods," in *Proc. of the Audio Engineering Society (AES)*, 2009.

[14] M. Müller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes," in *Proc. ICASSP*, vol. 0, pp. 1877–1880, 2009.

[15] J. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[16] N. C. Maddage, "Automatic structure detection for popular music," *IEEE Multimedia*, vol. 13, pp. 65–77, 2006.

[17] J. Reed, Y. Ueda, S. M. Siniscalchi, Y. Uchiyama, S. Sagayama, and C.-H. Lee, "Minimum classification error training to improve isolated chord recognition," in *Proc. ISMIR*, pp. 609–614, 2009.

[18] K. Lee and M. Slaney, "Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, pp. 291–301, 2008.

[19] C. Harte, M. Sandler, S. Abdallah, and E. Gómez, "Symbolic representation of musical chords: a proposed syntax for text annotations.," in *Proc. ISMIR*, 2005.

[20] A. Weller, D. Ellis, and T. Jebara, "Structured prediction models for chord transcription of music audio," in *Proc. ICMLA*, vol. 0, pp. 590–595, 2009.

LYRICS-TO-AUDIO ALIGNMENT AND PHRASE-LEVEL SEGMENTATION USING INCOMPLETE INTERNET-STYLE CHORD ANNOTATIONS

Matthias Mauch **Hiromasa Fujihara** **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{m.mauch, h.fujihara, m.goto}@aist.go.jp

ABSTRACT

We propose two novel lyrics-to-audio alignment methods which make use of additional chord information. In the first method we extend an existing hidden Markov model (HMM) for lyrics alignment [1] by adding a chord model based on the *chroma* features often used in automatic audio chord detection. However, the textual transcriptions found on the Internet usually provide chords only for the first among all verses (or choruses, etc.). The second method we propose is therefore designed to work on these incomplete transcriptions by finding a phrase-level segmentation of the song using the partial chord information available. This segmentation is then used to constrain the lyrics alignment. Both methods are tested against hand-labelled ground truth annotations of word beginnings. We use our first method to show that chords and lyrics complement each other, boosting accuracy from 59.1% (only chroma feature) and 46.0% (only phoneme feature) to 88.0% (0.51 seconds mean absolute displacement). Alignment performance decreases with incomplete chord annotations, but we show that our second method compensates for this information loss and achieves an accuracy of 72.7%.

1. INTRODUCTION

Few things can rival the importance of lyrics to the character and success of popular songs. Words and music come together to tell a story or to evoke a particular mood. Even musically untrained listeners can relate to situations and feelings described in the lyrics, and as a result very few hit songs are entirely instrumental [2]. Provided with the recording of a song and the corresponding lyrics transcript, a human listener can easily find out which position in the recording corresponds to a certain word. We call detecting these relationships by means of a computer program *lyrics-to-audio alignment*, a music computing task that has so far been solved only partially. Solutions to the problem have a wide range of commercial applications such as the computer-aided generation of annotations for karaoke, song-browsing by lyrics, and the generation of audio thumbnails [3], also known as audio summarization.

The first system addressing the lyrics-to-audio alignment problem was a multimodal approach proposed by Wang *et al.* [4], which has since been developed further [5]. The method makes relatively strong assumptions on the form and meter (time signature) of the songs, which enables preliminary chorus-detection and beat-tracking steps to aid the final low-level lyrics alignment step. In [1] a left-to-right hidden Markov model (HMM) architecture is used to align lyrics to audio, based on observed mel frequency cepstral coefficients (MFCCs). Here too, several preprocessing steps such as singing melody segregation and vocal activity detection are employed, but these make fewer assumptions, and the more complex HMM models the evolution of single phonemes in time. Similar HMM-based approaches have been used in [6] and [7]. A special case of lyrics alignment based on the speech melody of Cantonese has been presented in [8]. These existing lyrics-to-audio alignment systems have used only two information sources: the audio file and the lyrics.

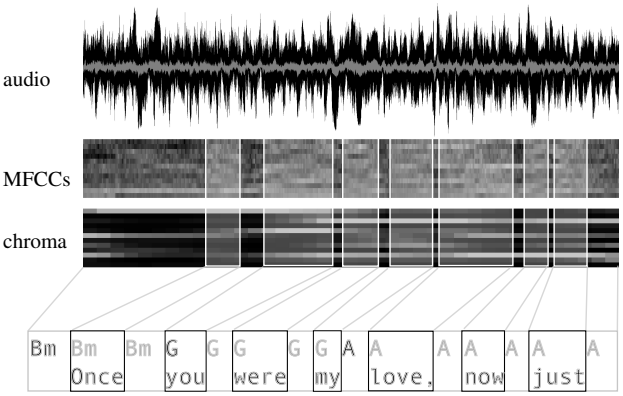


Figure 1: Integrating chord information in the lyrics-to-audio alignment process (schematic illustration). The chords printed black represent chord changes, grey chords are continued from a prior chord change. Word-chord combinations are aligned with two audio features: an MFCC-based phoneme feature and chroma.

In this paper we propose two novel techniques that integrate additional textual chord information (Figure 1) into the alignment framework:

1. an extension of the lyrics-to-audio alignment paradigm to incorporate chords and chroma features in the ideal case of complete chord information, and

2. a three-step method that can recover missing chord information by locating phrase-level boundaries based on the partially given chords.

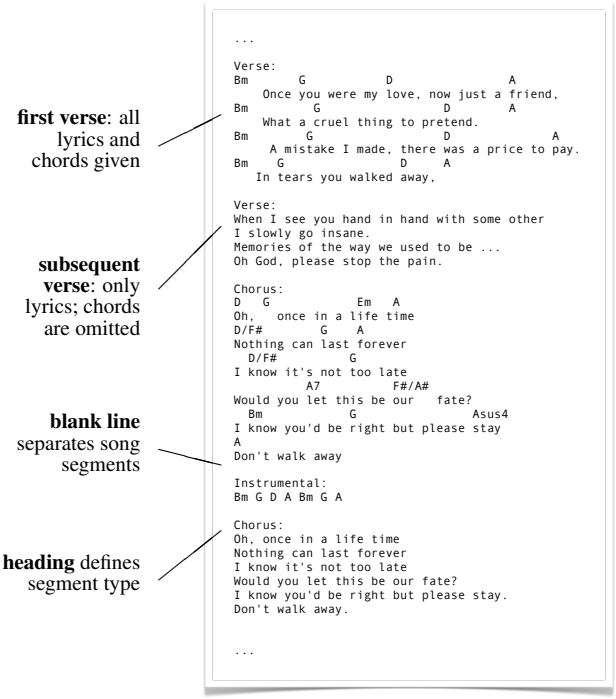


Figure 2: Excerpt adapted from "Once In A Lifetime" (RWC-MDB-P-2001 No. 82 [9]) in the chords and lyrics format similar to that found in many transcriptions on the Internet.

The motivation for the integration of chords is the vast availability of paired textual chord and lyrics transcriptions on the Internet through websites such as "Ultimate Guitar" [1] and "Chordie" [2] . Though there is no formal definition of the format used in the transcriptions appearing on the Internet, they will generally look similar to the one shown in Figure 2. It contains the lyrics of the song with chord labels written in the line above the corresponding lyrics line. Chords are usually written exactly over the words they start on, and labels written over whitespace denote chords that start before the next word. In our example (Figure 2) the lyrics of the verses are all accompanied by the same chord sequence, but the chord labels are only given for the first instance. This shortcut can be applied to any song segment type that has more than one instance, and transcribers usually use the shorter format to save space and effort. Song segment names can be indicated above the first line of the corresponding lyrics block. Song segments are separated by blank lines, and instrumental parts are given as a single line containing only the chord progression.

The rest of the paper is structured as follows. Section 2 describes the hidden Markov model we use for lyrics alignment and also provides the results in the case of complete chord information. Section 3 deals with the method that compensates for incomplete chord annotations by locating phrase-level boundaries, and discusses its results. Future work is discussed in Section 4, and Section 5 concludes the paper.

[1] http://www.ultimate-guitar.com

[2] http://www.chordie.com

2. USING CHORDS TO AID LYRICS-TO-AUDIO ALIGNMENT

This section presents our technique to align audio recordings with textual chord and lyrics transcriptions such as the ones described in Section 1. To show that the chord information does indeed aid lyrics alignment, we start with the case in which complete chord information is given. More precisely, we make the following assumptions:

complete lyrics Repeated lyrics are explicitly given.

segment names The names of song segments (e.g. *verse, chorus, ...*) are given above every lyrics block.

complete chords Chords for every song segment instance are given.

This last assumption is a departure from the format shown in Figure 2, and in Section 3 we will show that it can be relaxed.

An existing HMM-based lyrics alignment system is used as a baseline and then adapted for the additional input of 12-dimensional chroma features using an existing chord model [10]. We will give a short outline of the baseline method (Section 2.1), and then explain the extension to chroma and chords in Section 2.2. The results of the technique used in this section are given in Section 2.3.

2.1 Baseline Method

The baseline method [1] is based on a hidden Markov model (HMM) in which each phoneme is represented by three hidden states, and the observed nodes correspond to the low-level feature, which we will call *phoneme feature*. To be precise, given a phoneme state s, the 25 elements of the phoneme feature vector x_{m} with the distribution $P_{\mathrm{m}}(x_{\mathrm{m}}|s)$ consist of 12 MFCCs, 12 ΔMFCCs and 1 element containing the power difference (the subscript m stands for MFCC). These 12+12+1 elements are modelled as a 25-dimensional Gaussian mixture model with 16 mixture components. The transition probabilities between the three states of a phoneme and the Gaussian mixtures are trained on Japanese singing. For use with English lyrics, phonemes are retrieved using the Carnegie Mellon University Pronouncing Dictionary [3] and then mapped to their Japanese counterpart. A left-to-right layout is used for the HMM, i.e. all words appear in exactly the order provided. The possibility of pauses between words is modelled by introducing optional "short pause" states, whose phoneme feature emissions are trained from the non-voiced parts of the songs.

Since the main lyrics are usually present only in the predominant voice, the audio is pre-processed to eliminate all other sounds. To achieve this the main melody voice is segregated in three steps: first, the predominant fundamental frequency is detected using PreFEst [11]. The

[3] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

	phoneme	chroma
sampling	16000 Hz	11025 Hz
frame length	25ms	372ms
window	Hamming	Hamming
frame rate	100 Hz	10 Hz

Table 1: Signal processing parameters of the two audio features.

estimated frequency at every 10ms frame is used to find the further harmonics, and the weights of the harmonics are computed. Finally, the resulting harmonic structure is used to re-synthesize the segregated melody line. The MFCCs necessary for the inference are extracted from the re-synthesized voice at intervals of 10ms (details in Table 1).

A second pre-processing step is the vocal activity detection (VAD), which uses a simple probabilistic model with only two states (vocal or non-vocal) to find sung sections. The audio features used for this method are LPC-derived cepstral coefficients and ΔF0 (fundamental frequency difference). The method is parameterized such that few vocal regions are missed, even if this causes some instrumental regions to be misclassified as vocal.

The HMM is decoded using the Viterbi algorithm, during which the regions classified as non-vocal are constrained to emit only short pause states. This HMM is also a flexible framework which enables the integration of different features, as we explain below.

2.2 HMM Network with Lyrics and Chords

In order to integrate chords in the baseline method described above we need to parse the chords and lyrics files, calculate a low-level harmonic feature (chromagram) and extend the HMM so that it can process the additional information.

After parsing the chords and lyrics file of a song, every word can be associated with a chord, the lyrics line it is in, and the song segment this line is part of. In the present implementation a chord change on a word is assumed to start at the beginning of a word. While only the word-chord association is needed for the HMM, the line and segment information retained can later be used to obtain the locations of lines and song segments.

Chroma is a low-level feature that relates to musical harmony and has been used in many chord and key detection tasks [12, 13], but only rarely for chord alignment [14]. Chroma is also frequently used for score-to-audio alignment [15]. A chroma vector usually has twelve dimensions, containing activation values of the twelve pitch classes C, C#,. . ., B. Our chroma extraction method [16] uses the original audio before melody segregation. It first calculates a pitch spectrum with three bins per semitone, which is then adjusted for minor deviations from the standard 440 Hz tuning. Then, the background spectrum (local mean) is subtracted and the remaining spectrum is further normalized by the running standard deviation, which is a form of spectral whitening. Finally, assuming tones with an exponential harmonics envelope, the non-negative least squares algorithm [17] is used to find the activation of every note, which is then mapped to the corresponding chroma bin. Since chords change much more slowly than phonemes, the chroma method extracts features at a frame rate of 10Hz (Table 1), and to match the 100Hz rate of the MFCCs we duplicate the chroma vectors accordingly.

The hidden states of the HMM are designed as in the baseline method, with the difference that every state now has two properties: the phoneme and the chord. The observed emissions of the joint phoneme and chroma feature $x = (x_{\mathrm{m}}, x_{\mathrm{m}})$ are modelled by a probability distribution with log-density

$$\log P(x|s) = a \log P_{\mathrm{m}}(x_{\mathrm{m}}|s) + b \log P_{\mathrm{c}}(x_{\mathrm{c}}|s), \qquad (1)$$

where $P_m(x_m|s)$ is the baseline phoneme model, and $a = b = 1.0$ are weight parameters, which can be modified for testing. The subscripts m and c stand for MFCCs and chroma, respectively. $P_{\mathrm{c}}(x_{\mathrm{c}}|s)$ is the chord model, a set of 145 12-dimensional Gaussians that models the emissions of 145 different chords: 12 chord types (major, minor, major 7$^{\mathrm{th}}$,...) transposed to all 12 semitones, and one "no chord" type. The means of chord pitch classes are set to 1, all others to 0. All variance parameters in the diagonal covariance matrices are set to 0.2 [18].

For inference we use an implementation of the Viterbi algorithm developed for the baseline method. The output of the Viterbi decoder assigns to every phoneme the estimated time interval within the song.

2.3 Results I

We ran two sets of eight experiments, one with vocal activity detection, and one without. In each set we varied the phoneme and chroma feature weights a and b in (1) within $\{0.0,\ 0.5,\ 1.0\}$ (the case in which both features have zero weight is omitted). In order to evaluate the performance of our method we chose 15 songs (13 pop hits and 2 songs [4] from the RWC Music Database [9]) with English lyrics (see Table 2) and hand-labelled every word in these songs with its onset time in seconds. Previous work in lyrics-to-audio alignment has been evaluated only on phrase level, for which hand-labelling is less laborious, but the often uneven distribution of words over a lyric line makes word-level alignment a more meaningful ground truth representation. We evaluate the alignment according to two criteria. The mean percentage

$$p = \frac{1}{N_{\text{songs}}} \sum_{\text{song } k} \underbrace{\frac{1}{N_{\text{words}}} \sum_{\text{word } i} \mathbf{1}_{|\hat{t}_i - t_i| < 1} \times 100}_{\text{average percentage over } k^{\text{th}} \text{ song}} \qquad (2)$$

of start time estimates $\hat{t}_i$ that fall within one second of the start time t_i of the corresponding ground truth word, averaged over songs. We will simply call this measure *accu-*

[4] RWC-MDB-P-2001 Nos. 82 and 84.

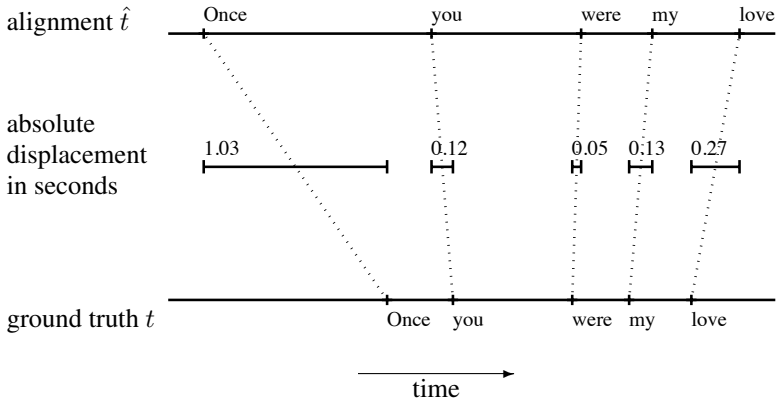


Figure 3: Calculation of the performance metrics. In this example, the accuracy p from Equation (2) is 80% because four of the five words have an absolute displacement of < 1 second. The mean absolute displacement d, see Equation (3), is 0.32 seconds, which is simply the arithmetic mean of the five absolute displacements.

	Artist	Song
1	Bangles	Eternal Flame
2	U2	With Or Without You
3	Robert Palmer	Addicted To Love
4	Martika	Toy Soldiers
5	Queen	We Are The Champions
6	Simon and Garfunkel	Cecilia
7	Otis Redding	The Dock Of The Bay
8	Shinya Iguchi (RWC)	Once In A Life Time
9	ABBA	Knowing Me Knowing You
10	Duran Duran	Ordinary World
11	Toto	Africa
12	Santana	Black Magic Woman
13	Shinya Iguchi (RWC)	Someday
14	Franz Ferdinand	Do You Want To
15	Duffy	Warwick Avenue

Table 2: The songs used for evaluation.

racy. A second measure is the mean absolute displacement

$$d = \frac{1}{N_{\text{songs}}} \sum_{\text{song } k} \underbrace{\frac{1}{N_{\text{words}}} \sum_{\text{word } i} |\hat{t}_i - t_i|}_{\text{mean abs. displacement in } k^{\text{th}} \text{ song}} \qquad (3)$$

between the time instant t_i at beginning of the i^{th} target word and its estimate $\hat{t}_i$, which is also averaged over all songs. Both metrics are illustrated in Figure 3.

Table 3 shows accuracy values, i.e. the mean percentage p as defined in Equation (2), for all tests. They can be summarized as follows: the highest accuracy of 88.0% is achieved with chroma and phoneme feature, but no vocal activity detection (VAD); the chroma features provide large scale alignment while the phonemes provide short-term alignment; VAD does improve results for the baseline method, i.e. when chroma is switched off. The next paragraphs provide more detailed explanations.

Table 3a shows results without VAD. The first column contains the results for which no chroma information was used ($b = 0.0$). The results in the second and third columns show that the additional information provided by the chord labels substantially boosts accuracy, for example from 38.4% ($a = 1.0$, $b = 0.0$) to the best result of 88.0% ($a = 1.0$, $b = 1.0$). While chord information yields the greatest improvement, we can also observe that using chord information alone does not provide a very good alignment result. For example, consider a chroma weight fixed at $b = 1.0$: when the phoneme feature is "off" ($a = 0.0$), we obtain only a mediocre alignment result of 59.1% accuracy, but setting $a = 1.0$, we obtain the top result of 88.0%. Our interpretation of these results is intuitive. Since chords occupy longer time spans and are therefore "hard to miss", they provide large scale alignment. The phonemes in the lyrics, on the other hand, often have very short time spans and are easy to miss in a rich polyphonic music environment. However, with good large scale alignment—as provided by the chords—their short term lyrics-to-audio alignment capabilities exceed those of chords.

We can also observe that when no chord information is given, VAD provides a similar kind of large scale alignment: for the baseline method ($a = 1.0$, $b = 0.0$) the use of VAD increases accuracy from 38.4% (Table 3a) to 46.0% (Table 3b). Table 3b also shows that using chroma and VAD together results in accuracy values slightly lower than the top ones, which has been caused by regions erroneously classified as non-vocal by VAD. The conclusion is: if full chord information is not available, use VAD; if chord information is available, use chroma instead.

The same pattern emerges for the mean absolute displacement d of words (Table 4). Here also, the best value, 0.51 seconds, is achieved when using both chroma and phoneme features (without VAD), compared to 1.26 seconds for the best result of the baseline method (with VAD).

One downside to the best methods mentioned so far is that the *complete chords* assumption is not always fulfilled, since transcribers often omit chord annotations for harmonically repeated sections (see Figure 2). The next section presents a method which—through intelligent use of the remaining chord information—deals with this situation and achieves results approaching the best ones seen above.

		chroma weight		
		$b = 0.0$	$b = 0.5$	$b = 1.0$
phoneme weight	$a = 0.0$	—	59.1	59.1
	$a = 0.5$	34.8	86.8	83.1
	$a = 1.0$	38.4	87.6	**88.0**

(a) accuracy without VAD

		chroma weight		
		$b = 0.0$	$b = 0.5$	$b = 1.0$
phoneme weight	$a = 0.0$	—	52.3	52.3
	$a = 0.5$	42.0	77.8	77.3
	$a = 1.0$	46.0	**81.9**	78.5

(b) accuracy with VAD

Table 3: **Accuracy:** mean percentage p as defined in Equation (2) for tests without and with vocal activity detection (VAD). Chroma and phoneme feature combined lead to the best results for the method using complete chord information. For a detailed discussion see Section 2.3.

		chroma weight		
		$b = 0.0$	$b = 0.5$	$b = 1.0$
phoneme weight	$a = 0.0$	—	1.98	1.99
	$a = 0.5$	8.22	0.63	1.06
	$a = 1.0$	6.93	0.72	**0.51**

(a) mean absolute displacement without VAD

		chroma weight		
		$b = 0.0$	$b = 0.5$	$b = 1.0$
phoneme weight	$a = 0.0$	—	3.74	3.73
	$a = 0.5$	5.52	1.67	1.69
	$a = 1.0$	4.67	**1.26**	1.65

(b) mean absolute displacement with VAD

Table 4: **Mean absolute displacement** d in seconds as defined in Equation (3) for tests without and with vocal activity detection (VAD). For a detailed discussion see Section 2.3.

3. RECOVERING PARTIALLY MISSING CHORDS

As we have seen in Figure 2, among all verses (or choruses, etc.) it is usually only the first one that is annotated with chords. Our method presented above cannot be applied directly anymore because in the remaining segments it is no longer clear which chord to associate with which word.

We will now consider this more difficult case by relaxing the "complete chords" assumption given in Section 2 and replace it with an assumption that is more in line with real world files:

incomplete chords Chords are given for the first occurrence of a song segment; subsequent occurrences of the same segment type have no chord information. They do still have the same number of lyric lines.

Transcriptions such as the one shown in Figure 2 now comply with our new set of assumptions.

While our basic method from Section 2 works in a single alignment step, the recovery method proposed in this section consists of the following three steps:

(a) naïve alignment: the basic alignment method as in Section 2, but missing chords are modelled by a "no chord" profile.

(b) phrase-level segmentation: the results of the alignment are used to build a new chord-based HMM for phrase-level segmentation.

(c) constrained alignment: the phrase-level segmentation result is fed back to the original alignment HMM: inference is performed constrained by phrase location.

Sections 3.1 to 3.3 will explain these steps in more detail and Section 3.4 presents the results.

3.1 Naïve Alignment

In this context we call "naïve" taking the best performing model from Section 2 ($a = 1.0$, $b = 1.0$, no VAD), which depends on chords and chroma, and apply it to the case of incomplete chords. We simply use the "no chord" model for words with missing chords, which ensures that no preference is given to any chord. This is step (a) in the above method outline. As could be expected, the scarcity of information leads to a substantial performance decrease, from 88.0% (as discussed in the previous section) to 58.44%. Clearly, the partial chord information is not sufficient to maintain a good long-term alignment over the whole song. However, the first occurrence of a song segment type such as a verse is always given with lyrics and chord information, and we have shown in Section 2.3 that alignment performance is generally good when both features are used, so it would be likely to find good alignment at least in the song segments for which chord information is not omitted. This is indeed the case: if we restrict the evaluation of the "naïvely" obtained results to the song segments annotated with chords, we obtain a higher level of accuracy: 72.1%. This has motivated us to implement the following two steps (b) and (c).

3.2 Phrase-level Segmentation

This is step (b), according to the system overview given above. We build a new HMM based entirely on chords and chroma, with three hierarchical levels depicted in Figure 4: chord, song segment, and song, based on the first (naïve) alignment step explained above. We assume indeed that in segments with complete chord information the word time estimates are nearly correct. Since the words are associated with chords, they provide us with an estimate of the chord lengths for every segment type. For each segment with

complete chords we will use these chord lengths to specify a new segment-specific HMM as a left-to-right chord sequence. Chords that cross a lyric line boundary, as the one from the first to the second lyric line in Figure 2, are duplicated such that a line always starts with a chord. This is important because otherwise the model based only on chroma observations could not find the correct new phrase beginning.

The model of each chord is determined by its length ℓ in seconds: it is modelled by $\lceil 2\ell \rceil$ states, i.e. two states per second. Only self-transitions or transitions to the next state are allowed (see Figure 4a). The self-transition probability is $s = 0.2$, and hence the expected duration of one state is 0.5 seconds at a frame rate of 10 Hz [5] . The expected duration of the chord is $\lceil \ell \rceil$, i.e. the length estimated in the previous step, up to rounding. Of course, we could have modelled each chord with one state with a higher self transition probability, but that would have led to a geometrically distributed chord duration model and hence to a bias towards short durations. The chord duration in our implementation model follows a negative binomial distribution—similar to the one used in [19]—in which the probability of very short chord durations is reduced.

The chord models are then concatenated to form a left-to-right model of the chord progression in one segment, as illustrated in Figure 4b. The segment HMMs are then combined to the final left-to-right song HMM. Since we assume we know the names of all segments, and hence their succession, we can simply concatenate the individual segment HMMs in the correct order, as can be seen in the example in Figure 4c. Of course, segment models may appear several times.

3.3 Constrained Alignment

This third step (c) integrates the results calculated in the two previous steps (a) and (b). We now run the HMM inference from the first step once again, but using the estimated lyric line beginnings from the previous step (b): we constrain the Viterbi search at frames of line beginnings to exactly the word the line starts with. To be precise, it is the short pause state preceding this word that is fixed to start at the estimated line beginning, so that lines that start with a chord, but no lyrics are not forced to start with a word.

3.4 Results II

We chose the method that performed best in the experiments reported in Section 2.3 for the further experiments, i.e. the feature weight parameters are set to $a = 1.0$ and $b = 1.0$. We used the same eight songs (see Table 2) and performed two more experiments, firstly the naïve application (a) of the original chord and lyrics alignment method, and secondly the full method including steps (a), (b) and (c). The accuracy results are given in Figure 5, together with the result obtained under complete chord information. First of all, we observe that the results of the naïve method are worse than the results with complete chord information: with respect to the method with complete chords at the top of the figure, removing chord information clearly decreases accuracy (defined in Equation 2) from 88.0% to 58.4%. Our proposed method, i.e. steps (a) to (c), can recover much of the lost information by applying phrase constraints, resulting in an accuracy of 72.7%.

[5] Since this HMM does not involve MFCCs we use the native chroma frame rate.

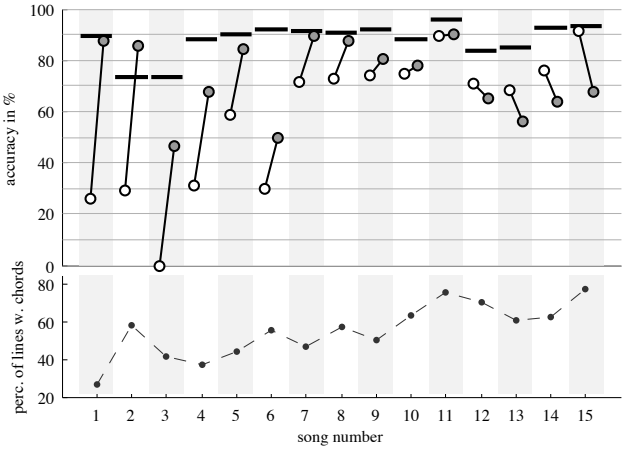


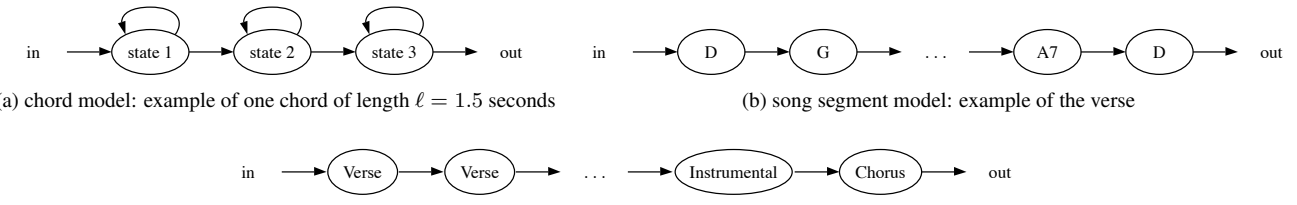
Figure 5: Songwise comparison. Top figure: black bars represent the accuracy obtained using all chord information (see Section 2); blank circles represent accuracy with partial chord data (chords of repeated segments removed); filled circles represent accuracy using our chord information recovery method as explained in Section 3. Bottom figure: proportion of lyrics lines for which chord information is available after partial removal.

method	accuracy in %
full chord information	88.0
incomplete chords (naïve method)	58.4
incomplete chords with recovery	72.7

Table 5: Accuracy for the methods presented in Section 3, as explained in 3.4.

Figure 5 illustrates where the chord information recovery method presented in this section works best: the blank and filled circles connected by solid lines show the improvement from the naïve method (blank) to the method with chord information recovery (filled). The songs are sorted by amount of improvement (same order as given in Table 2), and we observe that the recovery method improves results for the first 11 (of 15) songs. The differences are more substantial, if the accuracy of the naïve method is low. It is also interesting to observe that the improvement correlates negatively with the amount of chord information provided (see the percentage of lines with available chord information at the bottom of Figure 5).

Our results imply furthermore that the segmentation achieved in step (b) is very good. As far as we know, this is the first time that a chord progression model of song segments has been applied for song segmentation, made possible by the partially given chord data. A particularly interesting feature is the capability of finding structures down to the phrase level, as the example Figure 6 demonstrates.



(a) chord model: example of one chord of length $\ell = 1.5$ seconds

(b) song segment model: example of the verse

(c) song structure model

Figure 4: HMM for phrase-level segmentation. Though the network is a strict left-to-right HMM, it can be thought of in terms of three hierarchical layers representing chords, song segments, and song structure.

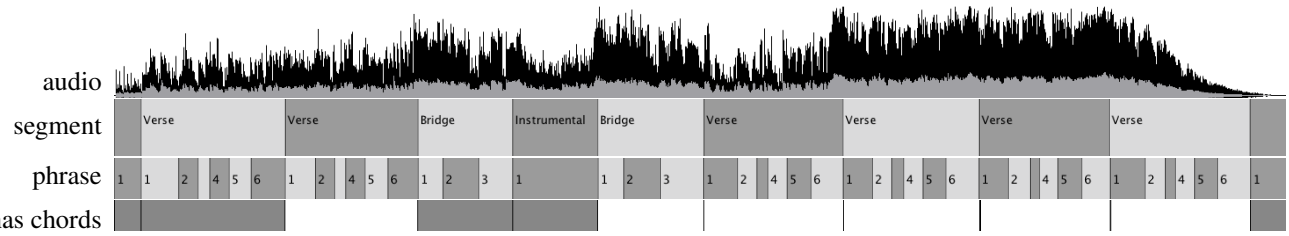


Figure 6: Automatic segmentation as explained in Section 3.2. The top line is a representation of the audio waveform of the song *Eternal Flame* by the Bangles, with means and maxima of positive values indicated in grey and black, respectively. Below are the automatically detected segments, with names from the chords and lyrics annotation file. Underneath is the corresponding phrase-level segmentation (i.e. lyric lines). We can clearly see that the verse has six lines, the bridge has only three, while the instrumental section has no lyrics and hence no further segmentation. In the bottom line the segments for which chord information was available are shaded dark.

4. DISCUSSION AND FUTURE WORK

Clearly, our chord information recovery method does not improve results for all songs, but in our experiments it did improve results for the majority of songs. No high-level music computing method can claim perfect accuracy, and systems that contain a number of successive steps suffer from errors that are propagated down to subsequent steps. We have presented such a system in this paper and are aware of this shortcoming. An approach that integrates all three steps into one would be much more elegant—and probably more effective. The main problem under partially missing chord data is that three song representations have to be aligned: lyrics, chords and audio. Finding a model that encompasses all poses a significant challenge and we are not aware of standard statistical models that directly lend themselves to this task. Once found, such a model could also provide more accurate alignment for cases in which the chord labelling is of low quality, e.g. when chords are not written exactly over the right words.

In a more efficient system, the audio feature generation should be unified to avoid the overhead of operating with different sample rates. We also plan to implement an application of the methods presented here: a machine that automatically generates guitar/singing karaoke annotations and allows a new advanced karaoke experience for musicians. In fact, real-time alignment could make such an application an electronic lead-sheet tool for bands. In the present study the chord and lyrics files were checked and edited so they could be parsed unambiguously. For example, we made sure that the names of song segments were unambiguously recognizable as such so they would not be parsed as lyrics. In an application aimed at non-expert users, this "clean-up" would have to be performed automatically, i.e. the parsing of the files would have to be much more robust. This is an interesting research problem in itself.

However, our primary goal is to further relax the assumptions made about the chord and lyrics data. For example, dropping the requirement that the succession or names of the song segments are given would make our method even more applicable to "sloppy" real world song annotations, and the segmentation method based on chord progressions presented in Section 3.3 is a promising starting point to finding song segments with no *a priori* information about their order. The next great music computing challenge is then to perform the task of chord-aided alignment without any prior chord information and automatically generate Internet-style chord and lyrics transcriptions. Though several systems for fully automatic structural segmentation and chord extraction exist, we are aware of none that combine the different parts needed: integrating more musical features is however one of the goals of the Sound and Music Computing Roadmap [6] and we expect that the most interesting music computing applications of the future will be those that aim to reach that goal.

5. CONCLUSIONS

This paper has shown that additional chord information in a textual "Internet" format can lead to substantially improved lyrics-to-chord alignment performance. This is true in the case in which chord information is provided for every part of the song, but also if the chords are only transcribed once for every song segment type (e.g. for the first of three verses), a shortcut often found in files in

[6] `http://smcnetwork.org/roadmap`

the Internet. We have proposed two methods that allow us to deal with these situations: the first one is based on an existing hidden Markov model that uses MFCC phoneme features for lyrics-to-audio alignment. We extend it by integrating chroma emissions and describe each hidden state in terms of the phoneme *and* the chord. We achieve an accuracy of 88.0% compared to 46.0% without chroma and 59.1% without phoneme features. If parts of the chord information are removed, the method performs worse (58.4%), though still better than the baseline method without chroma features. Our second proposed method succeeds in recovering much of the information lost: it uses the remaining partial chord information to build a new HMM with chord progression models for every song segment. Viterbi decoding of this HMM identifies the phrase structure of the song, so that lyrics alignment can be constrained to the correct phrase. This strategy boosts accuracy by more than 14 percentage points to 72.7%. We show that the improvement on individual songs is particularly marked when large parts of the chord information are missing.

We have noted that the results of the second method imply a good performance of the segmentation method. This is the first time that segment-specific chord progression models have been used for segmentation and phrase-finding. Similar models may allow us to further relax assumptions on the chords and lyrics input format and hence to achieve robust performance in real-world situations.

6. ACKNOWLEDGEMENTS

We would like to thank Anssi Klapuri and Gaël Richard for their ideas. This research was supported by CrestMuse, CREST, JST.

7. REFERENCES

[1] H. Fujihara, M. Goto, J. Ogata, K. Komatani, T. Ogata, and H. Okuno, "Automatic synchronization between lyrics and music CD recordings based on Viterbi alignment of segregated vocal signals," in *8th IEEE International Symposium on Multimedia (ISM'06)*, pp. 257–264, 2006.

[2] F. Bronson, *The Billboard Book of Number One Hits*. Billboard Books, 1997.

[3] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 96–104, 2005.

[4] Y. Wang, M.-Y. Kan, T. L. Nwe, A. Shenoy, and J. Yin, "LyricAlly: Automatic synchronization of acoustic musical signals and textual lyrics," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pp. 212–219, 2004.

[5] M.-Y. Kan, Y. Wang, D. Iskandar, T. L. Nwe, and A. Shenoy, "LyricAlly: Automatic synchronization of textual lyrics to acoustic music signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 338–349, 2008.

[6] H. Fujihara and M. Goto, "Three techniques for improving automatic synchronization between music and lyrics: Fricative detection, filler model, and novel feature vectors for vocal activity detection," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2008)*, 2008.

[7] A. Mesaros and T. Virtanen, "Automatic alignment of music audio and lyrics," in *Proceedings of the 11th Int. Conference on Digital Audio Effects (DAFx-08)*, 2008.

[8] C. H. Wong, W. M. Szeto, and K. H. Wong, "Automatic lyrics alignment for cantonese popular music," *Multimedia Systems*, vol. 12, no. 4–5, pp. 307–323, 2007.

[9] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, classical, and jazz music databases," in *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pp. 287–288, 2002.

[10] M. Mauch and S. Dixon, "Simultaneous estimation of chords and musical context from audio," *to appear in IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[11] M. Goto, "A real-time music scene description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication (ISCA Journal)*, vol. 43, no. 4, pp. 311–329, 2004.

[12] T. Fujishima, "Real time chord recognition of musical sound: a system using Common Lisp Music," in *Proceedings of the International Computer Music Conference (ICMC 1999)*, pp. 464–467, 1999.

[13] L. Oudre, Y. Grenier, and C. Févotte, "Template-based chord recognition: Influence of the chord types," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pp. 153–158, 2009.

[14] A. Sheh and D. Ellis, "Chord segmentation and recognition using EM-trained hidden Markov models," in *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003.

[15] R. Dannenberg and N. Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proceedings of the International Computer Music Conference (ICMC 2003)*, pp. 27–34, 2003.

[16] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[17] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, ch. 23. Prentice-Hall, 1974.

[18] M. Mauch, *Automatic Chord Transcription from Audio Using Computational Models of Musical Context*. PhD thesis, Queen Mary University of London, 2010.

[19] M. Mauch and S. Dixon, "A discrete mixture model for chord labelling," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pp. 45–50, 2008.

CHORD SEQUENCE PATTERNS IN OWL

Jens Wissmann
FZI Research Center for Information Technologies, Karlsruhe, Germany
jens.wissmann@fzi.de

Tillman Weyde
City University London, London, United Kingdom
t.e.weyde@city.ac.uk

Darrell Conklin
Department of Computer Science and AI Universidad del País Vasco, San Sebastián, Spain
IKERBASQUE, Basque Foundation for Science
darrell_conklin@ehu.es

ABSTRACT

Chord symbols and progressions are a common way to describe musical harmony. In this paper we present $\mathcal{SEQ}$, a pattern representation using the Web Ontology Language OWL DL and its application to modelling chord sequences. $\mathcal{SEQ}$ provides a logical representation of order information, which is not available directly in OWL DL, together with an intuitive notation. It therefore allows the use of OWL reasoners for tasks such as classification of sequences by patterns and determining subsumption relationships between the patterns. The $\mathcal{SEQ}$ representation is used to express distinctive pattern obtained using data mining of multiple viewpoints of chord sequences.

1. INTRODUCTION

The Semantic Web is an effort to augment the conventional Web with explicit machine-processable semantic metadata to serve as a backbone for a variety of automated content processing and retrieval task [1, 2]. In this context, several techniques for the logical description and querying of web data have been developed. Particularly, modelling of knowledge in web ontologies using the Description Logic OWL DL [3] enables automatic reasoning. However, these techniques have been developed with the focus on terminological metadata and the use of these techniques to reason on structured objects such as found in music representation is still in its beginnings.

For our approach, we chose chord sequences as a starting point as these are a popular representation and have increasingly gained research interest [4, 5]. They are also at a convenient and powerful level of musical abstraction. For example, within the “Music Ontology" effort patterns have been learned from chord sequences available in the Semantic Web data format RDF [6, 7]. The patterns themselves however have not been expressed with Semantic Web techniques. Indeed, neither RDF nor OWL offer ad hoc support for representing sequential structures.

We have developed a generic representation for sequential patterns in OWL DL that we call $\mathcal{SEQ}$, extending the work of [8], and applied it to chord sequence representation. Notation and expressivity are similar to regular expressions, and allow the expression of different levels of abstraction. Several reasoning tasks on such a representation can be solved using readily available OWL reasoners. In a web retrieval scenario, for example, instance checking can be used to find chord sequences that match or contain a search pattern. More interestingly, subsumption checking analyses pattern inclusion.

To demonstrate how the $\mathcal{SEQ}$ representation can be used to enrich the results of pattern discovery, we translated distinctive chord patterns, which were learned from a corpus using a statistical learning approach in [9], into $\mathcal{SEQ}$ and used an OWL reasoner for the calculation of subsumption relations and instance retrieval.

2. MODELLING KNOWLEDGE IN OWL DL

OWL DL belongs to the Description Logic (DL) family of knowledge representation languages [10]. DLs are popular for describing the knowledge of a domain of interest by formalising its terminology using

- *instances* $i, j, \ldots$,
- *concepts* $C, D, \ldots$ and
- *properties* $R, S, \ldots$

Most DLs correspond to fragments of first order logic such that *instances*, *concepts* and *properties* correspond to *constants*, *unary predicates* and *binary predicates*.

An *ontology* is a set of axioms that define relationships between these terms. The part of the ontology that asserts facts about instances is called the ABox, while the part that defines the terminology is called TBox. From a first order logic perspective, ABox axioms assert predicates on constants while TBox axioms describe predicate structures on variables. Basic forms of terminological axioms are

- concept subsumption ($C \sqsubseteq D$) and
- equivalence ($C \equiv D$).

Basic forms of assertional axioms are

- type assertions ($i \in C$) and
- property assertions ($R(i, j)$).

Here C and D can stand for *atomic* concepts but can also be composite *expressions* as we will further illustrate.

In this paper we mainly focus on modelling structural aspects of chord sequences, but will consider some example concept expressions from the domain of music metadata as DL syntax was originally introduced for describing terminologies and it is therefore most intuitive to describe

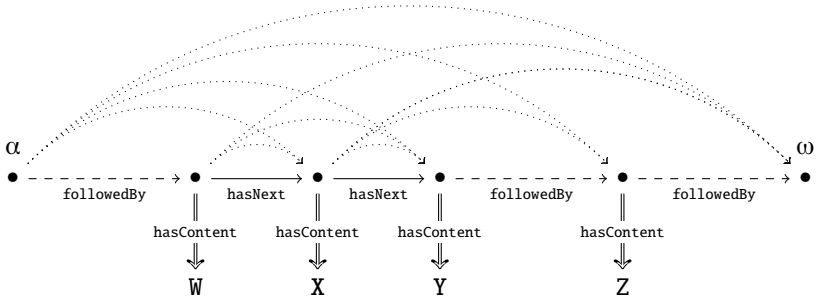


Figure 1: Structure of an example sequential pattern

the relationships between words. A motivation for this is also to highlight the possibilities of DLs for reasoning on musical structures and musical metadata within one single logical framework. For example, consider the TBox

$$\texttt{Musician} \equiv \exists \texttt{performed}.\texttt{Music} \sqcup \exists \texttt{wrote}.\texttt{Music}$$
$$\texttt{Composer} \equiv \exists \texttt{wrote}.\texttt{Music}$$

These axioms define a musician as somebody who performed or wrote music, and a composer to be someone who wrote music. Here boolean constructs and property restrictions are used to form expressions. DLs provide boolean constructors $\neg C$, $C \sqcap D$, $C \sqcup D$. As DLs have first order logic semantics we can think of these as complement, intersection and union of sets (of instances). Further, DLs allow to quantify over properties ($\exists R.C$, $\forall R.C$, $\exists^{=n} R.C$, $\exists^{\leq n} R.C$, $\exists^{\geq n} R.C$), e.g. stating that for an instance that is a `Composer` there exists a property `wrote` with the range `Music`. OWL Reasoners provide certain standard reasoning services. For example, by *subsumption reasoning* on the TBox a reasoner can be infer that all composers are necessarily musicians (`Composer` $\sqsubseteq$ `Musician`). In fact all subsumption problems in DLs are decidable, i.e. we can do this for any two concept descriptions. So the main challenge is to capture the interesting aspects of a terminology as DL axioms, whereas the reasoning is done automatically.

A further reasoner task is classification of an ABox with respect to TBox concepts. Consider the facts

$$\texttt{wrote}(\texttt{mozart}, \texttt{magic\_flute}),$$
$$\texttt{wrote}(\texttt{shakespeare}, \texttt{hamlet}),$$
$$\texttt{magic\_flute} \in \texttt{Music},$$
$$\texttt{shakespeare} \in \forall \texttt{wrote}.\texttt{Literature}$$

Here, for example, Mozart will be classified as composer and musician. Shakespeare will not be classified as musician as he just wrote literature.

Additional DL constructs exist that allow to assert subproperty relationship, inverse property relationship and characteristics of properties such as being functional, transitive, reflexive, irreflexive, symmetric or asymmetric. We refer the reader to [10] and [11] for a more detailed discussion of DLs.

3. MODELLING SEQUENCES IN OWL

The wish to model sequences arises naturally in the music domain, given its temporal nature. Unfortunately, there are no native constructs within OWL DL to express sequence patterns. Drummond et al. [8] proposed to use a *linked list* approach. We extended this approach and developed $\mathcal{SEQ}$, an ontological representation of sequence patterns.

In the following we describe the axiomatisation of basic $\mathcal{SEQ}$ patterns and give examples. The axiomatization of the linked list structure follows the ideas of Drummond et al. [8]. One difference is that we introduce an initial component because this is crucial for the behaviour of pattern subsumption and for the creation of more complex pattern constructs. Further we introduce a notation to express sequences in a more intuitive (yet formal) way.

The core structure of a $\mathcal{SEQ}$ pattern is similar to a linked list. Figure 1 shows an example. Components of patterns are linked by solid dots. Each component can be associated with linking and content properties: Linking is expressed by using the *functional* property `hasNext` (solid arrow) that connects a component to its immediate successor or by using the *transitive* property `followedBy` (dashed arrow) that connects a component to all following components. A *pattern* is characterised by restricting these properties. As the subproperty relationship `hasNext` $\sqsubseteq$ `followedBy` is asserted for $\mathcal{SEQ}$ patterns, `followedBy` relationships are are implicitly defined between all connected components (dotted arrows).

The property `hasContent` can be used to describe the content of a pattern component. Finally, we introduce an intital component (α) with no precursor and no content and a final component (ω) with no successor and no content (see table 1 for definitions) . A sequence pattern $\texttt{SP}_1$ that describes sequences that consist of "some instances of `W`, then `X`, then `Y`, then followed by `Z`" (as shown in fig. 1) can be described by the DL concept

$$\begin{aligned}\texttt{SP}_1 \equiv \alpha \sqcap \exists \texttt{followedBy}.(&\exists \texttt{hasContent}.\texttt{W} \\ \sqcap \quad &\exists \texttt{hasNext}.(\exists \texttt{hasContent}.\texttt{X} \\ \sqcap \quad &\exists \texttt{hasNext}.(\exists \texttt{hasContent}.\texttt{Y} \\ \sqcap &\exists \texttt{followedBy}.(\exists \texttt{hasContent}.\texttt{Z} \\ \sqcap &\exists \texttt{followedBy}.\omega))))\end{aligned}$$

For simplification, we can state this expression equivalently

	Syntax	Semantics	
succeeds	$C \triangleright D$	$C \sqcap \exists \texttt{hasNext}.D$	TBox
follows	$C \cdots D$	$C \sqcap \exists \texttt{isFollowedBy}.D$	
has content	$[C]$	$\exists \texttt{hasContent}.C$	
initial	α	$\neg\exists \texttt{followedBy}^{-}.\top \sqcap \exists^{\leq 0}\texttt{hasContent}.\top$	
terminal	ω	$\neg\exists \texttt{followedBy}.\top \sqcap \exists^{\leq 0}\texttt{hasContent}.\top$	

Table 1: A selection of $\mathcal{SEQ}$ constructs and their definition. C and D denote arbitrary DL concepts

in $\mathcal{SEQ}$ notation as

$$\texttt{SP}_1 \equiv [\texttt{W}] \triangleright [\texttt{X}] \triangleright [\texttt{Y}] \cdots [\texttt{Z}]$$

with α and ω are not explicitly stated and arrows, dots and square brackets capturing the details of the succeeds, follows and content restrictions. Consider, the pattern

$$\texttt{SP}_2 \equiv [\texttt{W}] \triangleright [\texttt{X}] \triangleright [\texttt{Y}] \triangleright [\texttt{Z}]$$

The difference with $\texttt{SP}_1$ here is that Y has to be directly followed by Z. Intuitively we expect that $\texttt{SP}_2$ is more specialized than $\texttt{SP}_1$ and all instances of $\texttt{SP}_2$ will also be instances of $\texttt{SP}_1$. As we have formalized $\mathcal{SEQ}$ patterns as DL concepts, we can directly use the machinery for computing DL concept subsumption to automatically compute pattern subsumption. In this case a standard DL reasoner will infer the subsumption relationship $\texttt{SP}_2 \sqsubseteq \texttt{SP}_1$ (taking into consideration that hasNext is a subproperty of followedBy).

The possibilities of subsumption reasoning get more interesting when we use concept *expressions* (such as we have done in the Musician example) rather than simple concept names. For example we could define a chord by its properties, e.g.

$$\begin{bmatrix} \exists \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \\ \sqcap \exists \ \texttt{seventh} \ . \ \texttt{b7} \end{bmatrix}$$

where the pattern characterises a chord by the properties root, triad and seventh. Given another more general pattern that for example only restricts root and triad a reasoner could infer a subsumption relationship such as

$$\begin{bmatrix} \exists \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \\ \sqcap \exists \ \texttt{seventh} \ . \ \texttt{b7} \end{bmatrix} \sqsubseteq \begin{bmatrix} \exists \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \end{bmatrix}$$

In the work described in the following section we restrict ourselves to patterns that describe their content as a conjunction of features (functional properties) as we can discover patterns of this form automatically using the pattern discovery method by [9]. Note, that in principle it is also possible to make use of further DL operators when defining patterns. For example, the pattern

$$\begin{bmatrix} \exists \ \texttt{root} \ . \ \neg(\texttt{F} \sqcup \texttt{G}) \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \end{bmatrix}$$

matches major chords that have a root other then F or G, and given our previous example pattern would give rise to the subsumption relationship:

$$\begin{bmatrix} \exists \ \texttt{root} \ . \ \texttt{C} \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \\ \sqcap \exists \ \texttt{seventh} \ . \ \texttt{b7} \end{bmatrix} \sqsubseteq \begin{bmatrix} \exists \ \texttt{root} \ . \ \neg(\texttt{F} \sqcup \texttt{G}) \\ \sqcap \exists \ \texttt{triad} \ . \ \texttt{Maj} \end{bmatrix}$$

Naturally the question arises how such patterns can be created in practise. As manual modelling is often costly and time consuming, it is interesting to investigate methods for automatic pattern creation. In the following section we will outline the relationship of the $\mathcal{SEQ}$ formalism to the established viewpoint approach to automatic pattern discovery.

4. SUBSUMPTION STRUCTURE OF DISTINCTIVE CHORD PATTERNS

Though $\mathcal{SEQ}$ patterns can be specified in a top-down manner by a knowledge engineer, it is interesting to learn them from a corpus of music. This approach leads to the question which patterns are most relevant and interesting, which is a typical question from the field of data mining. Depending on the application, there are different relevant properties. For the classification of music, which is very useful in a Semantic Web scenario, we are interested in distinctive patterns that help differentiate one class from another, and general patterns that apply to many relevant data sets in a class. Conklin [9] has applied this approach to chord sequences and found a number of relevant patterns that we further analysed using $\mathcal{SEQ}$.

4.1 Representation of Feature Set Patterns

Pattern discovery using *multiple viewpoints* is a machine learning approach for discovering patterns in sequential musical data. It has mainly been used for discovery of patterns in melodies, but recently also for learning patterns in chord progressions [9]. Input and patterns are represented using a *feature set* representation [12].

For a sequence of musical events (e.g. chords), viewpoints are computed. A viewpoint τ is a function from events to values in a specific range set. A feature is defined as $\tau : v$ where τ is a feature name and v a feature value. A *feature set* then is a *conjunction* of features

$$\{\tau_1 : v_1, \ldots, \tau_n : v_n\}$$

and a *pattern* is a sequence

$$f_1, \ldots, f_m$$

where each f_i is a feature set.

	events:	Im7	IVm7	Im7	Vbm7b5	IV7	IIIsm7b5
features	degree	I	IV	I	Vb	IV	IIIs
	basedegree	I	IV	I	V	IV	III
	kp	I	II/IV	I	V/VII	II/IV	III
	triad	Min	Min	Min	Dim	Maj	Dim
	rootmvt	⊥	4n	5n	5b	7s	7n

Table 2: Example decomposition of chord-events into feature sets for viewpoint learning

Table 2 shows an example of how a chord progression is represented as a sequence of feature sets. The viewpoints `degree`, `triad` and `basedegree` directly relate to the chord symbol. Relationships between events are modelled as features that belong to a single event and have to be read as referring back to the previous event. The feature `rootmvt : 4n` for example expresses that the current root event is a fourth about the previous event. In the case of the first event features of this kind take the value ⊥ as there is no previous event they could refer to. We use further viewpoints in later examples such as `meeus` that indicates harmonic function (tonic (`T`), dominant (`D`) or subdominant (`S`)) as described by [13], `kp` that indicates chord degree classes as described by [14] and `ratio(dur)` that indicates the relative duration of an event.

4.2 Translation of Feature Set Patterns to $\mathcal{SEQ}$

Feature set patterns can be translated into $\mathcal{SEQ}$ using a translation function T that is defined as follows. Each feature $\tau : v$ can be translated into a DL property restriction

$$T(\tau : v) = \exists \tau.v$$

where every viewpoint τ corresponds to a *functional* property τ and the value v is the filler that the property is restricted to. A feature set is described by a DL concept intersection

$$T(\{\tau_1 : v_1, \ldots, \tau_n : v_n\}) = \exists \tau_1.v_1 \sqcap \ldots \sqcap \exists \tau_n.v_n$$

A feature set pattern $f_1 \ldots f_m$ can then be expressed using `hasNext` relationships as

$$T(f_1, \ldots, f_m) = [T(f_1)] \triangleright \ldots \triangleright [T(f_m)]$$

In the following we will show examples of genre-specific chord sequence patterns that have been learned from chord sequences tagged with the genres *jazz*, *classic* and *pop*.

4.3 Maximally General Distinctive Chord Patterns

A *maximally general distinctive pattern* (MGDP) is a pattern that is distinctive above a threshold and not subsumed by any other distinctive pattern. They are least likely to overfit the corpus and hence most likely to be useful for classification. To measure distinctiveness the likelihood ratio of a pattern P is employed. This is defined in [9, 15] as

$$\Delta(P) \stackrel{def}{=} \frac{p(P|\oplus)}{p(P|\ominus)} = \frac{c^{\oplus}(P) \times n^{\ominus}}{c^{\ominus}(P) \times n^{\oplus}}$$

where $p(P|\oplus)$ is the probability of the pattern P in the corpus, $p(P|\ominus)$ is the probability of the pattern P in the anticorpus (consisting of pieces of different classes), $c^{\oplus}(P)$ and $c^{\ominus}(P)$ are the count of the pattern in the corpus and the anticorpus respectively, and $n^{\oplus}$ and $n^{\ominus}$ are the size of the corpus and anticorpus respectively.

Figure 2 (top) illustrates three MGDPs chosen from a much larger set of highly distinctive patterns that were discovered in a corpus of 856 chord sequences, divided into genres jazz (338), classical (235), and popular (283) [16]. The interest $\Delta(P)$ of the pattern is indicated: for example, the first pattern is overrepresented by a factor of 12.45. The numbers in brackets indicate that the length of the pattern is 2 and it occurs in 65 jazz sequences but only 8 sequences in the anticorpus (classical and popular sequences). The pattern indicates a minor triad on degree `III`, followed by any triad on degree `III` (due to the fact that the `meeus` property indicates the `T` (tonic) chord transformation). Note that despite this high level of abstraction in this pattern it remains highly distinctive in this corpus for the jazz genre.

In the middle of Figure 2, instances of each of these patterns are represented as fully saturated feature set sequences.

4.4 Subsumption Structure

To compute the subsumption structure of the learned viewpoint patterns we translated viewpoint patterns into $\mathcal{SEQ}$ concepts and used a DL reasoner to infer their subsumption relationships.

The bottom part of Figure 2 illustrates a small fragment of a subsumption hierarchy of viewpoint patterns, created from a larger set of pattern that are *maximally general* and *distinctive* (MGDP). The subsumption relationships were computed by the $\mathcal{SEQ}$-translation of the MGDPs. To compute the subsumption relationships we translated the patterns into $\mathcal{SEQ}$ and then use the OWL reasoner Pellet [1] to classify. This figure has restricted the representation to five MGDP that appear on the righthand side of the hierarchy.

Some internal concepts have been constructed in $\mathcal{SEQ}$ and it can be seen how these capture commonalities between the MGDPs thereby providing richer structure to a flat MGDP set. At the left hand side of the figure are "primitive" features contained in single component patterns. Substantial structure can be seen. For example, `triad : Min` can be seen to occur in four MGDPs and in addition in one internal $\mathcal{SEQ}$ pattern.

[1] http://clarkparsia.com/pellet/

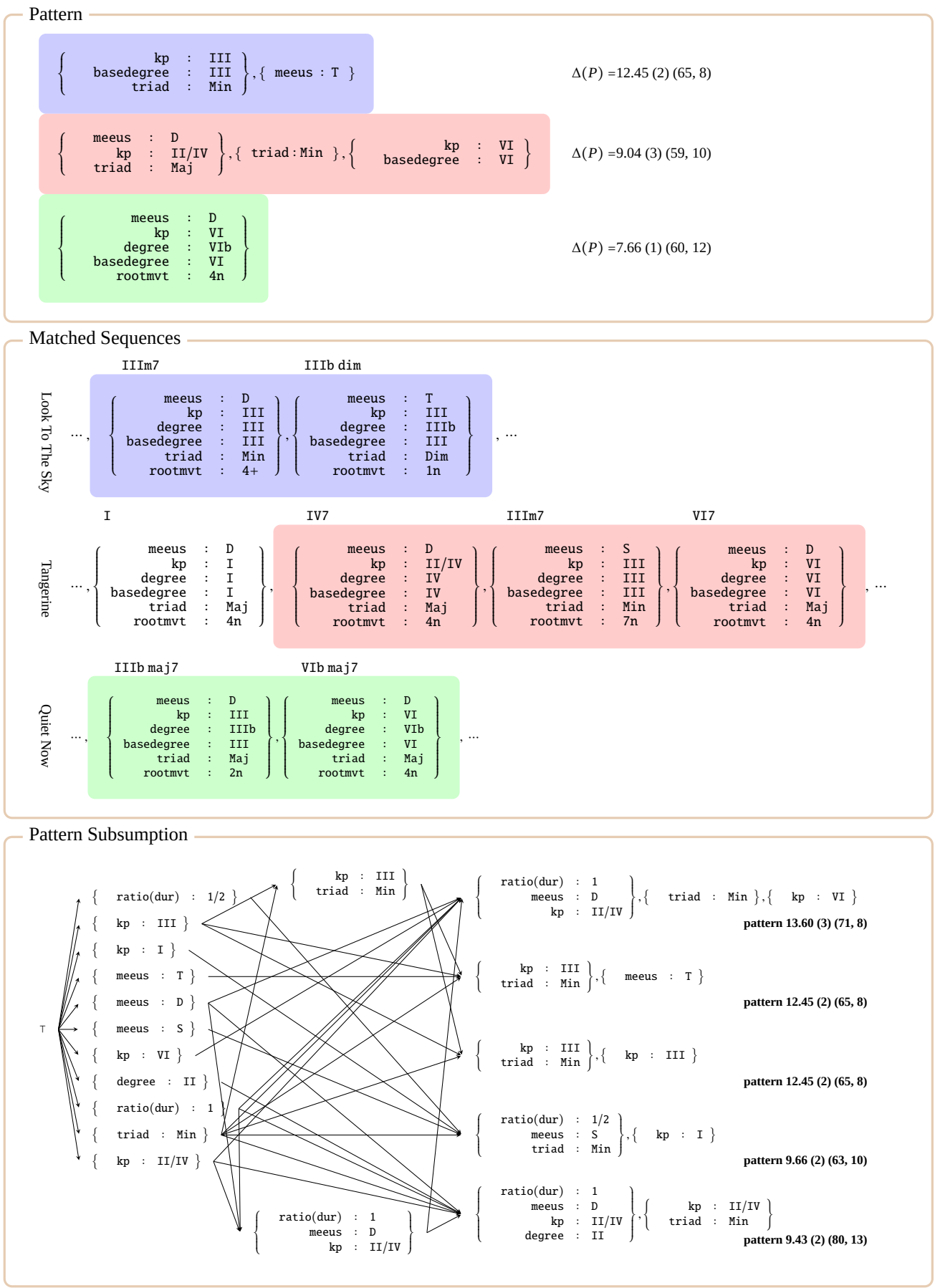


Figure 2: Example of learned MGDP-patterns (top), matching sequences (middle) and pattern subsumption (bottom).

5. CONCLUSIONS

We introduced the $\mathcal{SEQ}$ language and showed how it expresses sequential patterns and discussed some aspects of syntax and the DL semantics of $\mathcal{SEQ}$. We demonstrated the usage of $\mathcal{SEQ}$ to represent and analyse chord patterns that were discovered from a corpus using viewpoint learning. A DL reasoner can then use such patterns to classify instance data. Further, the patterns can be classified automatically in terms of their subsumption relationships as illustrated for distinctive patterns from [9].

Several possibilities for future research arise. Reasoning on metadata descriptions (as in our introductory example) and structural descriptions within the same reasoning formalism might offer interesting new application possibilities for musicology and music information retrieval. Further, the machine-learned descriptions could be complemented with relationships between basic musical entities such as notes, scales and chord as found in the harmony literature.

6. REFERENCES

[1] T. Berners-Lee, *Weaving the Web : the past, present and future of the World Wide Web by its inventor*. London: Orion Business, 1999.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The SemanticWeb," *Scientific American*, vol. 284, pp. 34–43, May 2001.

[3] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, eds., *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[4] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez, "Symbolic representation of musical chords: A proposed syntax for text annotations," in *ISMIR*, pp. 66–71, 2005.

[5] A. Sheh and D. P. W. Ellis, "Chord segmentation and recognition using em-trained hidden markov models.," in *ISMIR*, 2003.

[6] A. Anglade and S. Dixon, "Characterisation of harmony with inductive logic programming," in *Proc. of the Ninth International Conference on Music Information Retrieval (ISMIR)*, (Philadelphia, USA), pp. 63–68, Sep 2008.

[7] M. Mauch, S. Dixon, C. Harte, M. Casey, and B. Fields, "Discovering chord idioms through Beatles and Real Book songs," in *Proceedings of ISMIR 2007 Vienna, Austria*, pp. 255–258, 2007.

[8] N. Drummond, A. Rector, R. Stevens, G. Moulton, M. Horridge, H. H. Wang, and J. Seidenberg, "Putting OWL in Order: Patterns for Sequences in OWL," in *2nd OWL Experiences and Directions Workshop, Athens, GA*, 2006.

[9] D. Conklin, "Discovery of distinctive patterns in music," To appear in *Intelligent Data Analysis*, vol. 14, no. 5, 2010.

[10] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[11] F. Baader, I. Horrocks, and U. Sattler, "Description Logics," in *Handbook of Knowledge Representation* (F. van Harmelen, V. Lifschitz, and B. Porter, eds.), Elsevier, 2007.

[12] D. Conklin and M. Bergeron, "Feature set patterns in music," *Computer Music Journal*, vol. 32, no. 1, pp. 60–70, 2008.

[13] N. Meeus, "Toward a post-Schoenbergian grammar of tonal and pre-tonal harmonic progressions," *Music Theory Online*, vol. 6, January 2000.

[14] S. Kostka and D. Payne, *Tonal Harmony*. McGraw-Hill, 2003.

[15] D. Conklin, "Distinctive Patterns in the First Movement of Brahms's String Quartet in C Minor," To appear in *Journal of Mathematics and Music*, vol. 4, no. 2, 2010.

[16] C. Pérez-Sancho, D. Rizo, and J.-M. Iñesta, "Genre classification using chords and stochastic language models," *Connection Science*, vol. 20, no. 2&3, pp. 145–159, 2009.

PERFORMANCE RENDERING FOR POLYPHONIC PIANO MUSIC WITH A COMBINATION OF PROBABILISTIC MODELS FOR MELODY AND HARMONY

Tae Hun Kim, Satoru Fukayama, Takuya Nishimoto, Shigeki Sagayama
Graduate School of Information Science and Technology
The University of Tokyo
{kim,fukayama,nishi,sagayama}@hil.t.u-tokyo.ac.jp

ABSTRACT

We present a method to generate human-like performance expression for polyphonic piano music. Probabilistic models and machine learning techniques have been successfully applied to solve the problem of generating human-like expressive performance, given a music score. In case of polyphonic music, however, it was difficult to make models tractable and a huge amount of training data was necessary, because performance contexts and relationships of performance expressions are very complex. To overcome these problems, we propose a method with a combination of probabilistic models for melody and harmony. The experimental results show that the proposed method was able to generate fluctuations of performance expression parameters for polyphonic piano music such like human performers do. The results of the subjective evaluations are also reported which indicate that their sounds were human-like and had certain degree of musicality.

1. INTRODUCTION

Human music performances include expression which is not written in scores. This is one of the reasons why people prefer performed music by famous performers rather than performances without expression which can be directly rendered from the score itself. But the mechanism of human music performances is still not clear and therefore it is very difficult to generate human-like music performance expression automatically, given a music score.

However, if we can construct such a system, it will be useful for general users to obtain a copyright-free music performance automatically which can be used as a background music for their own original videos and home pages, for instance. In addition, it will be also useful for supporting music composition and education for not only professional musicians but also general users who have little knowledge of music. For example, users can obtain human-like expressive performances for their original songs very easily, even if they can not play a music instrument.

We focus on piano performances because there are many solo pieces for piano and performance expressions can be represented with relatively few parameters comparing with string and wind instruments. If we look at human performance expression for polyphonic piano music, we can observe that tempo, dynamics and performed note durations are changing permanently for melodies (Fig. 1) and observe that differences of note onset-time, velocity and performed note duration for harmonies (Fig. 2) . The problem of generating human-like expressive performance is to estimate fluctuations of these performance expression parameters, given a music score. However, it is a very difficult problem, because many relationships between score and its performance expression are not explicit.

To solve this problem, probabilistic models and machine learning from human performance expression have been successfully applied. In case of polyphonic music, however, it was difficult to make models tractable and a huge amount of training data was necessary, because performance contexts and relationships of performance expressions are very complex.

In this paper, we present a method to generate human-like expressive performances for polyphonic piano music by learning from human performance expression while avoiding data sparseness problems.

2. RELATED WORKS

Many computational methods for automatic music performance rendering have been proposed, such as rule-based expert systems, query-by-case methods and machine learning [2]. In this section, only probabilistic model based works which take advantage of machine learning from human performance expression will be briefly summarized.

S. Flossmann introduces *performance context* and propose a probabilistic model for monophonic melody[3]. The model is trained with a large amount of human performance expression recorded by two professional pianists, N. Magaloff and R. Batik. Performance expression is predicted by estimation of 3 parameters such as tempo, dynamics and articulations. K. Teramura propose a computational method for imitating music performance expression of famous pianists using Gaussian Process with a monophonic melody model[4]. For predicting tempo fluctuations, she considers periodical characteristics of tempo and reports good results for pieces in three-four time such as waltzes.[5].

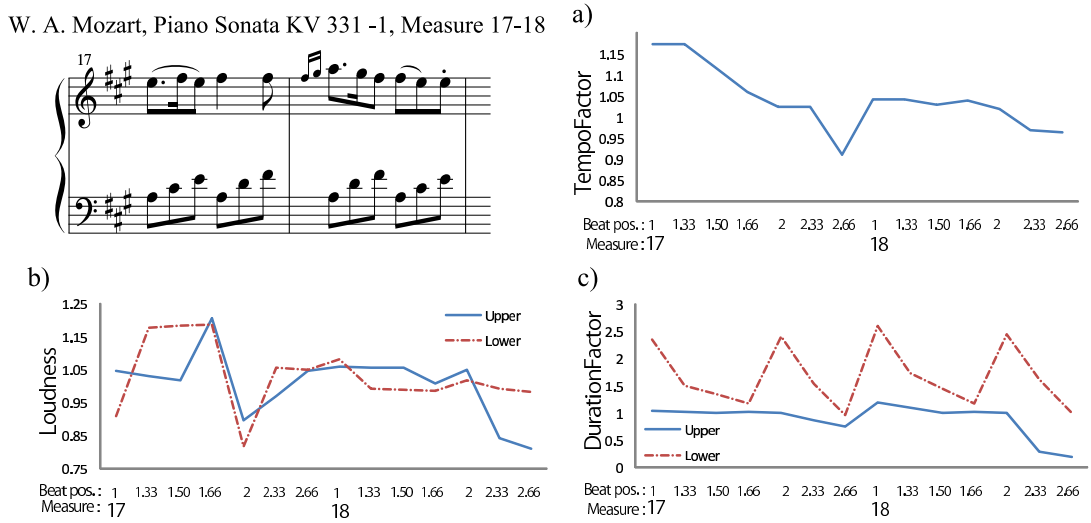


Figure 1. Performance expression for melody of a human performer (Ingrid Haebler, from CrestMusePEDB[1]). a) Fluctuations of instantaneous tempo which are calculated by the equation (1). b) Dynamics calculated by the equation (2). c) Fluctuations of performed duration which are calculated by the equation (3). These graphs show how human performance expression for melody looks like. Note that upper and lower outer-voices have different performance expressions.

These two works are based on monophonic melody models and they report quite good results for generating human-like performance expression. However, they don't discuss how to generate expression for polyphonic piano music.

There are some possible methods to generate performance expression for polyphonic piano music with monophonic melody models. For example, 1) generate performance expression of extracted main melody (e. g. soprano voice) from given music pieces and copy them to all other voices, 2) extract all voices and generate performance expression for each voice and combine them, 3) treat a polyphonic music piece as an one dimensional sequence of notes which is sorted by time and pitch orders.

However, these methods have some limitations. By 1) it is impossible to generate the characteristics of human music performance expression for polyphonic piano music which are mentioned above. 2) has a problem to extract all voices from given score which is very difficult. By 3) it is possible to generate different performance expression for each note even though input scores are polyphonic, however, its musical structure will be lost.

G. Grindlay proposes a Hidden Markov Model-based expressive music performance system and he discusses how to generate performance expression for accompaniment parts [6]. However, it is not possible to generate differences of performance expression of each note in case of harmony.

3. METHOD

In this paper, we present a method for performance rendering for polyphonic piano music with a combination of probabilistic models for melody and harmony.

Polyphonic piano music can be approximated with a combination of upper and lower outer-voices and harmonies. This is because human percepts outer-voices easier than inner-voices and inner-voices are related with sounds of harmonies[7]. In addition, upper and lower outer-voices don't have the same performance expression (Fig. 1).

Music performance can be regarded as a combination of global and local expressions. Global expression is the expression resulted from interpretation of expression marks such as *cresc.* and *rit.*. Local expression is the expression which has no expression marks for itself and is conditioned by local note-level contexts such as Fig. 1.

In this paper, we will focus on local expression, because we believe that it is important for *human-likeness* of performance expression. However, generating local expression is difficult, because relationships to their contexts are not explicit. To overcome this problem, a probabilistic model is applicable because it makes possible to capture some tendencies of relationships. If we assume that the relationships between contexts and performance expressions are probabilistic, then generating local expression can be regarded as an optimization problem to find the most probabilistic sequence of performance expression, given a sequence of performance contexts which are represented by rich score features.

Based on these discussions, we propose following strategy to generate human-like performance expression for polyphonic piano music:

Learning performance expression

1. Split music scores for training into right and left hands.
2. Extract sequences of the highest notes for right hand and sequences of the lowest notes for left hand. These two sequences are regarded as outer-voices. Extract harmonies for each hand.
3. Train left and right hand melody models with corresponding performance expressions of the extracted outer-voices.

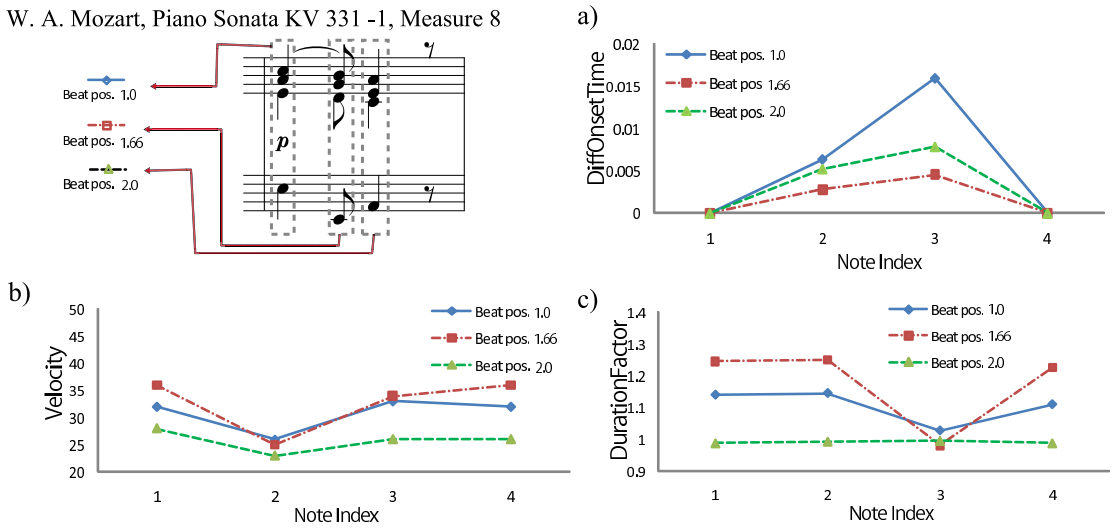


Figure 2. Performance expression for harmony of a human performer (Ingird Haebler, from CrestMusePEDB). Note index is an index of sorted sequence of notes in a given harmony (The first note is the highest note in a given harmony). a) Differences of onset-time calculated by the equation (4). b) Differences of velocity calculated by the equation (5). c) Differences of performed note duration calculated by the equation (6). These graphs show how human performs harmonies. Note that each note has a different performance expression to the others.

4. Train left and right harmony models with corresponding performance expressions of the extracted harmonies.

Generating performance expression

1. Split a input music score into right and left hands.
2. Extract a sequence of the highest notes and estimate its performance expression with the trained melody model for right hand. Extract a sequence of the lowest notes and estimate its performance expression with the trained melody model for left hand.
3. Estimate performance expression for right hand harmonies using the trained right hand harmony model and estimate performance expression for left hand harmonies using the trained left hand harmony model.
4. Combine the 4 estimated performance expressions.

On this strategy, we split scores into upper and lower outervoices and harmonies. Therefore, it is possible to generate human-like performance expression for polyphonic piano music with relatively simple score features and first-order Markov chain models using a small amount of training data.

Followings are details of melody and harmony model and its learning and estimation.

3.1 Melody model

3.1.1 Performance expression parameters

For melody model, 3 performance expression parameters are considered: instantaneous tempo, loudness and performed note duration. Fluctuations of each performance parameter can be modeled with first-order linear Markov chains if we assume that the current parameter value is conditioned only by its previous parameter value. In this way, we can avoid fast fluctuations which cause unnatural sounds.

Instantaneous Tempo

$$\text{TempoFactor}_t = \log\left(\frac{\text{Tempo}_t}{\text{Tempo}_{\text{avg}}^{\text{scope}}}\right) \quad (1)$$

where $\text{Tempo}_{\text{avg}}^{\text{scope}}$ is the average tempo of n_{t-3}, n_{t-2}, n_{t-1}, n_t, n_{t+1}, when n_t is the current note to perform.

Loudness

$$\text{Loudness}_t^{\text{melody}} = \log\left(\frac{\text{Velocity}_t}{\text{Velocity}_{\text{avg}}^{\text{scope}}}\right) \quad (2)$$

where $\text{Velocity}_{\text{avg}}^{\text{scope}}$ is the average velocity of n_{t-3}, n_{t-2}, n_{t-1}, n_t, n_{t+1}, when n_t is the current note to perform.

Performed note duration

$$\text{DurationFactor}_t^{\text{melody}} = \log\left(\frac{\text{Duration}_t^{\text{real}}}{\text{Duration}_t^{\text{score}}}\right) \quad (3)$$

where $\text{Duration}_t^{\text{score}}$ and $\text{Duration}_t^{\text{real}}$ are nominal durations in given scores and performed note durations by human performers, respectively.

3.1.2 Score features

For performance expression of a melody, we assume that performance contexts are different for each performance expression parameter, even if the performing note is identical. Therefore, different score features are considered for each parameter (Table 1).

Table 1. Score features for melody model

Instantaneous tempo	Loudness	Performed note duration
-	Pitch	-
-	$\text{Duration}^{\text{score}}$	$\text{Duration}^{\text{score}}$
NoteInterval I, II, III, IV	NoteInterval I, II, III, IV	NoteInterval I, II, III, IV
DurationRatio I, II, III, IV	DurationRatio I, II, III, IV	DurationRatio I, II, III, IV
Metric I, II	Metric I, II	Metric I, II
Articulation Marks	Articulation Marks	Articulation Marks

In the score features, Pitch means an absolute pitch as MIDI note number and $\text{Duration}^{\text{score}}$ is a nominal duration in a given score. NoteInterval I, II, III, IV are the note intervals of pair $(n_{t-3},\ n_{t-2})$, $(n_{t-2},\ n_{t-1})$, $(n_{t-1},\ n_t)$, (n_t, n_{t+1}) where n_t is the current note, respectively. DurationRatio I, II, III, IV are duration ratios of each pair above. Metric is a variable which has a value from $\{very\_strong, strong, weak\}$ and Metric I, II are Metric of previous and current note, respectively. ArticulationMarks is an articulation mark such as *staccato*,*accent* and *fermata*.

3.2 Harmony model

3.2.1 Performance expression parameters

For harmony model, 3 performance expression parameters are considered: difference of onset-time, loudness and performed note duration. A sequence of parameter values which is sorted by pitch can be modeled with a first-order linear Markov chain to avoid a large difference of parameter values which causes an unnatural sound.

Difference of onset-time

$$\text{DiffOnsetTime}_i = \text{OnsetTime}_o - \text{OnsetTime}_i \quad (4)$$

where OnsetTime_o is onset time of a note, which belongs to outer-voices, in a given harmony. OnsetTime_i is onset time of the current note to perform. DiffOnsetTime is a difference of onset-times, when a quarter note has a length of 1.0 (See [1]).

Loudness

$$\text{Loudness}_i^{\text{harmony}} = \log\left(\frac{\text{Velocity}_i}{\text{Velocity}_o}\right) \quad (5)$$

where Velocity_o is velocity of a note, which belongs to outer voices, in a given harmony. Velocity_i is velocity of the current note to perform.

Performed note duration

$$\text{DurationFactor}_i^{\text{harmony}} = \log\left(\frac{\text{Duration}_i^{\text{real}}}{\text{Duration}_o^{\text{real}}}\right) \quad (6)$$

Table 2. Score features for harmony model

Difference of onset-time	Loudness	Performed note duration
Pitch		
$\text{Duration}^{\text{score}}$		
NoteDistance		
OuterNote		

where $\text{Duration}_o^{\text{real}}$ and $\text{Duration}_i^{\text{real}}$ are performed duration of a note, which belongs to outer voices, in a given harmony and performed duration of the current note, respectively.

3.2.2 Score features

For performance expression of a harmony, same score features are considered for all 3 performance expression parameters (Table 2).

In score features, Pitch is an absolute pitch as MIDI note number and $\text{Duration}^{\text{score}}$ is a nominal duration in a score. NoteDistance is measured by a note interval between the note belongs to outer voices and the current note to perform. OuterNote is a variable which has *true*, if the current note is an outer note of the harmony and *false*, otherwise.

3.3 Learning and estimation

Because sequences of performance expression for both of melody and harmony are modeled by linear chain Markov models, any of HMM-like probabilistic models is applicable for learning and estimation. In the experiments, we employed Conditional Random Fields[8], which show better performances for input sequences represented by rich features. Both of melody and harmony models are trainable with Maximum Likelihood Estimation using Stochastic Gradient Descent algorithm[9] and performance expression can be estimated with Viterbi algorithm. To implement the proposed method, we used "crfsgd" package by León Bottou [1] .

3.4 Quantization of performance expression parameters

Performance expression parameters should be quantized into discrete values for learning and estimating performance expression, because CRFs are frameworks for predicting label sequences and therefore it is not able to estimate continuous values. In the experiments, the parameters were quantized into 32 labels with k-means algorithm. Initial values of the algorithm are given by random sampling from the prior distribution of performance expression parameters which is obtained from the training data and therefore a non-linear quantization preserving the prior distribution of performance parameter values is possible, for example, more probable values of performance expression parameters are quantized into small size bins.

[1] http://leon.bottou.org/projects/sgd

Table 3. Training data for experiment 1. 4 performances of 1 piece were used totally.

Piece	Performer
Piano Sonata KV331, 1st Mov.	Hiroko Nakamura
Piano Sonata KV331, 1st Mov.	Norio Shimizu
Piano Sonata KV331, 1st Mov.	Ingrid Haebler
Piano Sonata KV331, 1st Mov.	Lily Kraus

4. EVALUATION

To evaluate proposed method for generating human-like performance expression for polyphonic piano music, we conducted two experiments: for test pieces which are known to the implemented system and for test pieces which are unknown to the system. To evaluate human-likeness and musicality of the generated performance expression with the proposed method, we also conducted subjective evaluations for them.

In the experiments, we trained melody and harmony models with CrestMusePEDB ver. 2.3[1]. In the subjective evaluations, we used sound samples which are rendered with a sampling-based virtual instrument, "Garritan Instruments for Finale2009" from Garritan Libraries.

4.1 Experimental environments

Experiment 1 – For known pieces to the system In experiment 1, we evaluated the proposed method for known pieces to the system. Melody and harmony models were trained with 4 human performances of W. A. Mozart, Piano Sonata, KV. 331, 1st Movement (Table 3). As the test piece, we used the same piece. It is composed with several harmonies and therefore we can see, if the proposed method is able to generate performance expression for polyphonic piano music.

Experiment 2 – For unknown pieces to the system In experiment 2, we evaluated the proposed method for unknown pieces to the system. Melody and harmony models were trained with 14 pieces of F. Chopin which are performed by Vladimir Ashkenazy (Table 4). As the test piece, we used F. Chopin, Nocturne No. 10, Op. 32, 2nd Movement which is not included in the training data set. This piece was selected because melodies and harmonies are mixed and it is usually performed with profound expression.

4.2 Generation results

As the results of experiment 1, the common performance expression of 4 performers were learned and generated.

We mentioned that the upper and lower outer-voices have different fluctuations of performance expression to each other in case of human performances. Probably, this is because each voice has a different role, for example, the upper outer-voice is a melody and the lower outer-voice is an accompaniment. Fig. 3 shows that there are also differences between fluctuations of performance expression parameters such as loudness and performed note duration of upper and lower outer-voices in the generated performance expression with proposed method.

Table 4. Training data for experiment 2. 14 performances of 14 pieces were used totally.

Pieces	Performer
Prelude Op. 28 No. 1, 4, 7, 15, 20 (5 pieces)	V. Ashkenazy
Etude Op.10-3, 10-4, 25-11 (3 pieces)	V. Ashkenazy
Waltz Op. 18, 34-2, 64-2, 69-1, 69-2 (5 pieces)	V. Ashkenazy
Nocturne No. 2 Op. 9-2 (1 piece)	V. Ashkenazy

For harmony, we mentioned that human performance expression have different onset-time, loudness, and performed duration for each note. Probably, this is resulted by influences from the interpretation of the piece by performers and the characteristcs of their fingering. Fig. 4 shows that generated performance expressions with the proposed method also has different performance parameter values such as onset-time, loudness and performed duration for each note.

The fluctuations were not random, for example, performed durations of lower voice showed a certain pattern according to a given accompaniment pattern, for example, the lowest note A is performed as *legato*.

These results indicate that with the proposed method, it is able to automatically generate fluctuations of performance expression parameters for *known* polyphonic piano music, with certain degree of musicality.

From the results of experiment 2, we also can see that upper and lower outer-voices have different fluctuations of performance expression parameters. For harmony, all notes have different performance parameter values to each other (Fig. 5).

The fluctuations were not random, for example, tempo fluctuations showed a certain pattern according to measure borders (a tempo-arch was observed for each measure).

These results show that with proposed method, it is also able to automatically generate fluctuations of performance expression parameters for *unknown* polyphonic piano music, with certain degree of musicality.

4.3 Subjective evaluation

The experimental results show that the proposed method is able to generate meaningful fluctuations of performance expression parameters for polyphonic piano music. To evaluate their human-likeness and musicality, we have conducted subjective evaluations.

For the evaluations, we prepared 3 performance expressions which are generated with the proposed method: W. A. Mozart, Piano Sonata, KV. 331, 1st Movement which

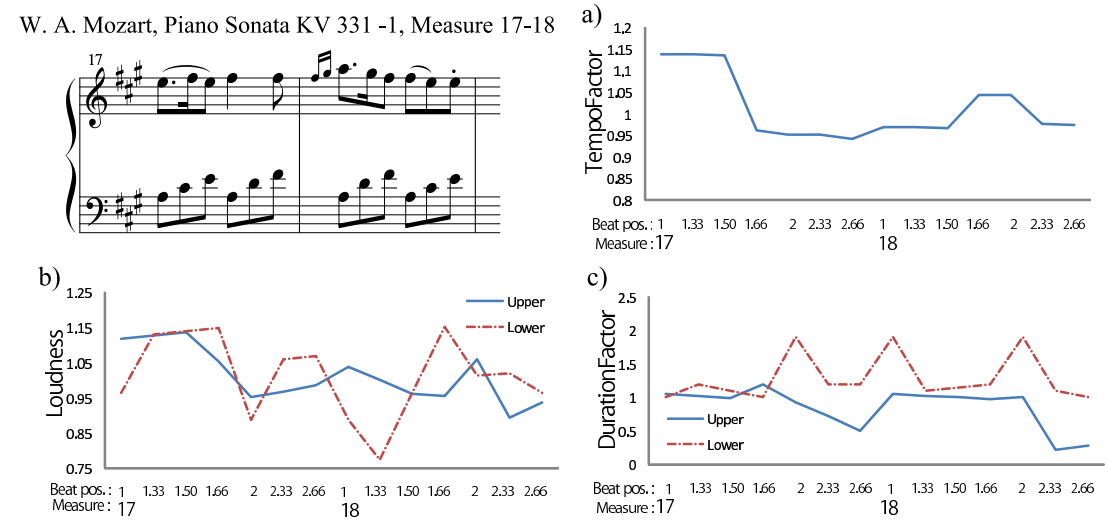


Figure 3. The results of experiment 1 (melody). a) Fluctuations of instantaneous tempo calculated by the equation (1). b) Dynamics calculated with the equations (2). c) Fluctuations of performed note duration which are calculated by the equation (3). These results show that the generated performance expression for melody with the proposed method have fluctuations of performance expression parameters such like human performance expression do and they are not arbitrary.

is the result of experiment 1, F. Chopin, Nocturne No. 10, Op. 32, 2nd Movement which is the result of experiment 2 and W. A. Mozart, Piano Sonata KV. 545, 3rd Movement whose performance expression is newly generated with the models trained with 6 pieces of Mozart's piano sonata [2] performed by M. J. Pires.

In addition, we prepared 3 more sound samples for each piece (total 12 samples): performance without expression, human performance expression, and performance expression for comparison. Performance expression for comparison has expression only for upper outer-voice and it is copied to the other voices and therefore upper and lower outer-voices have the same expression and each note of harmony also has the same expression. The purpose of preparing performance expression for comparison is to see if the proposed method considering polyphonic characteristics is effective to generate human-like performance expression for polyphonic piano music.

Human-likeness and musicality of each sound sample were evaluated by 25 participants [3] with 6 scaled scores, where 1 means "not human-like at all" and 6 means "very human-like" for human-likeness and 1 means "not musical at all" and 6 means "very musical" for musicality.

Fig. 6 shows the results of subjective evaluations. [4] In this figure, we can see that performance expressions generated with proposed method were evaluated that they sounded more human-like and musical for all 3 pieces comparing with performances without expression. In the cases of Mozart's Piano Sonata KV.331, 1 Mov. and Chopin's Nocturne No. 10, Op.32, 2nd Mov., participants evaluated them with the scores which are very closed to human performance expression (ANOVA indicates that thess differences are not significant). It means that performance expressions with the proposed method sounded human-like and musical for these pieces.

For Mozart's Piano Sonata, KV. 545, 3rd Mov., performance expression with proposed method obtained relatively low score comparing with human performance expression. This might be because for Mozart's piano sonata with fast tempo, global expression by interpretations of expression marks and musical structure is more important than local expression which are related with note-level contexts. In the experiment, human performance expression included both of local and global expressions and therefore performance expression with proposed method obtained such a low score comparing with human performance expression.

However, the averages of the 3 pieces show that performance expressions with proposed method obtained better scores than performance expressions for comparison and overall human-likeness and musicality of performance expressions with proposed method are most closed to human performance expressions comparing with other sound samples. Probably, this is because the proposed method is able to generate more profound expression than performance expression for comparison, especially for polyphonic piano music.

These results show that the proposed method is effective to generate performance expression for polyphonic piano music and its generation results sound human-like and have certain degree of musicality.

5. CONCLUSION

We proposed a method to generate human-like performances for polyphonic piano music with a combination of probabilistic melody and harmony models. With the experiments

[2] W. A. Mozart, Piano Sonata, KV279-1, 279-2, 279-3, 331-1, 545-1, 545-2. Note that KV545-3 is not included in the training data.

[3] 6 non-musicians, 17 hobby-musicians and 2 professional musicians participated in the experiment.

[4] Differences of average scores are tested by the Analysis Of Variance and its post-hoc test ($p < 0.05$)

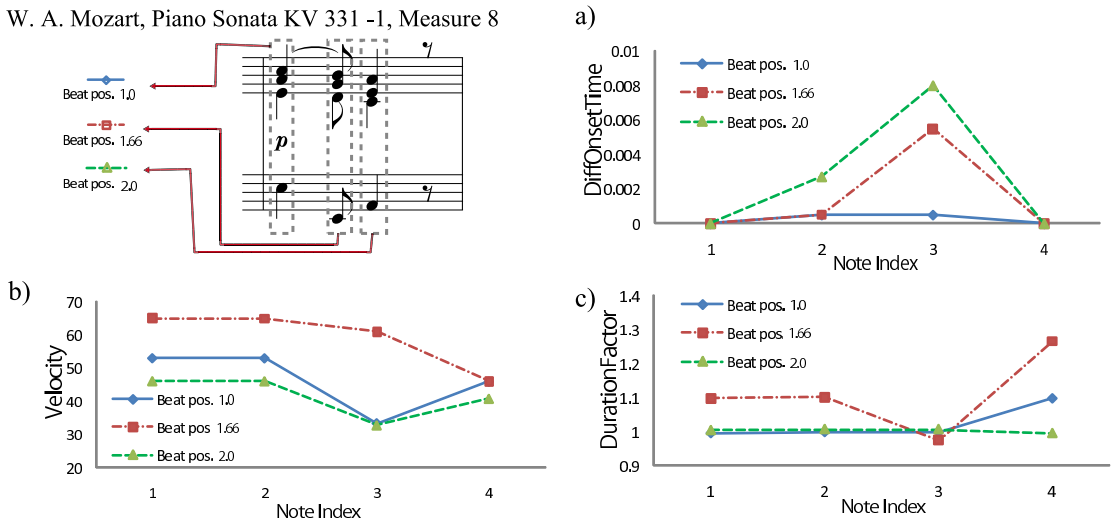


Figure 4. The results of experiment 1(harmony). a) Differences of onset-time calculated by the equation (4). b) Differences of velocity calculated by the equation (5). c) Differences of performed note duration which are calculated by the equation (6). These results show that the generated performance expression for harmony with the proposed method have different parameter values for each note, such like human performance expression do.

and the subjective evaluations, we showed that our method is effective to generate performance expression for known and unknown polyphonic piano music and they sound human-like and musical with profound expression.

Global expression by interpretations of expression marks and musical structure are also very important for human-like expressive performance. But expression marks are not easy to interpret, because each expression mark also has its certain performance context and therefore its interpretation is varying. As the next step, we will challenge to learn and estimate global expression to generate more human-like performances with more profound expression.

We believe that there is a possibility to learn personality of a specific performer through training models with his or her real performances. Therefore, we will experiment on the proposed method to see its ability to generate distinguish performance expression for each performer. This will be useful not only for searching a specific performer from a music database, but also for musicological researches of human music performances.

6. ACKNOWLEDGMENTS

This research is funded by CrestMuse Project [5] and a part of this research is supported by Samsung Scholarship Foundation [6].

7. REFERENCES

[1] M. Hashida, T. Matsui, and H. Katayose, "A new database describing deviation information of performance expressions," in *Proceedings of the 9th International Conference of Music Information Retrieval(ISMIR)*, 2008.

[2] A. Kirke and E. Miranda, "A survey of computer systems for expressive music performance," *ACM Comput. Surv.*, vol. 42, no. 1, 2009.

[3] S. Flossmann, M. Grachten, and G. Widmer, "Expressive performance rendering: Introducing performance context," in *Proceedings of the 6th Sound Music and Computing Conference(SMC)*, pp. 155–160, 2009.

[4] K. Teramura and H. Okuma, "Gaussian process regression for rendering music performance," in *Proceedings of the 10th International Conference on Music Perception and Cognition(ICMPC)*, 2008.

[5] K. Teramura and S. Maeda, "Statistical learning of tempo variations for imitating piano performance," *Information Processing Society Japan, Special Interst Group for Music and Computer, Technical Report*, vol. 85, no. 12, 2010. (Japanese).

[6] G. Grindlay and D. Helmhold, "Modeling, analyzing, and synthesizing expressive piano performance with graphical models," *Machine Learning*, vol. 65, pp. 361–387, 2006.

[7] R. Parncutt, "Accents and expression in piano performance," in *Perspektiven und Methoden einer Systemischen Musikwissenschaft* (K. W. Niemöller, ed.), Systemische Musikwissenschaft, pp. 163–185, Frankfurt am Main: Peter Lang, 2003.

[8] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: probabilistic models for segmenting and labeling sequence data," in *International Conference on Machine Learning*, pp. 282–289, 2001.

[9] L. Bottou, "Stochastic gradient learning in neural networks," in *Proceedings of Neuro-Nîmes 91*, (Nimes, France), EC2, 1991.

[5] http://www.crestmuse.jp/index-e.html
[6] http://www.ssscholarship.com

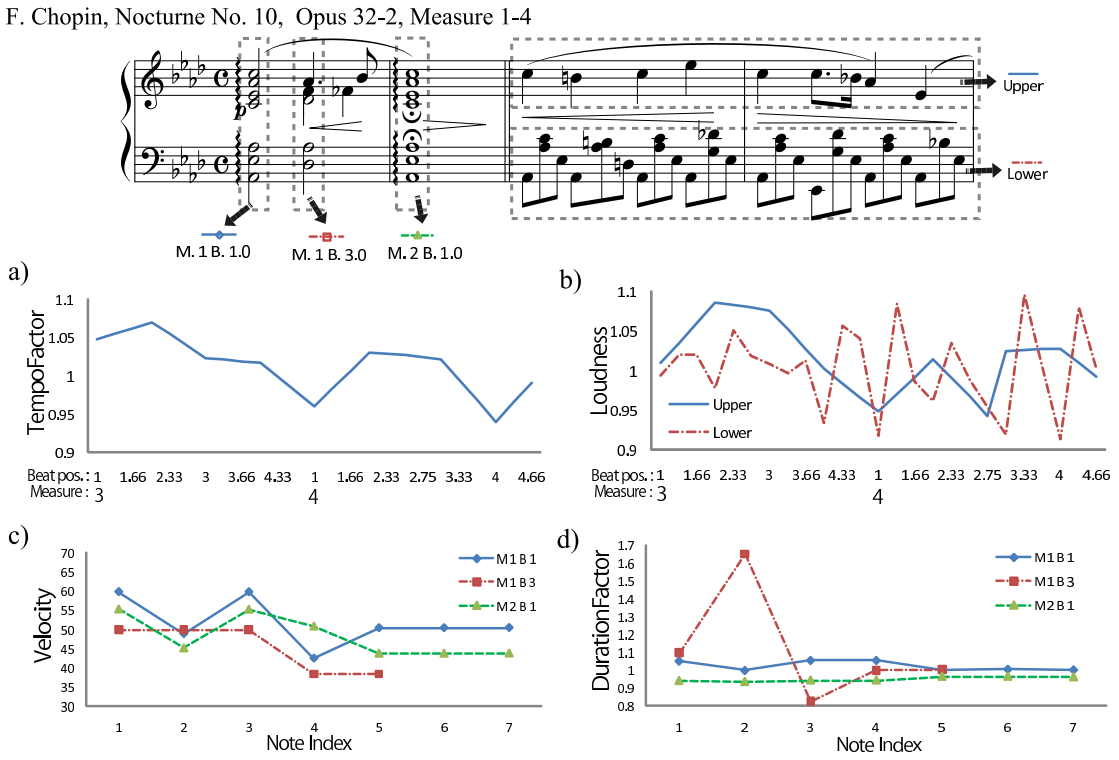


Figure 5. The results of experiment 2. a) Fluctuations of instantaneous tempo calculated by the equation (1). b) Dynamics for melody calculated by the equation (2). c) Velocity differences of the harmony notes which are calculated with the equation (5). d) Differences of performed note duration of the harmony notes which are calculated by the equation (6). Fluctuations of performed note duration for melody and Onset-time differences of the harmony notes are omitted due to space limitations.

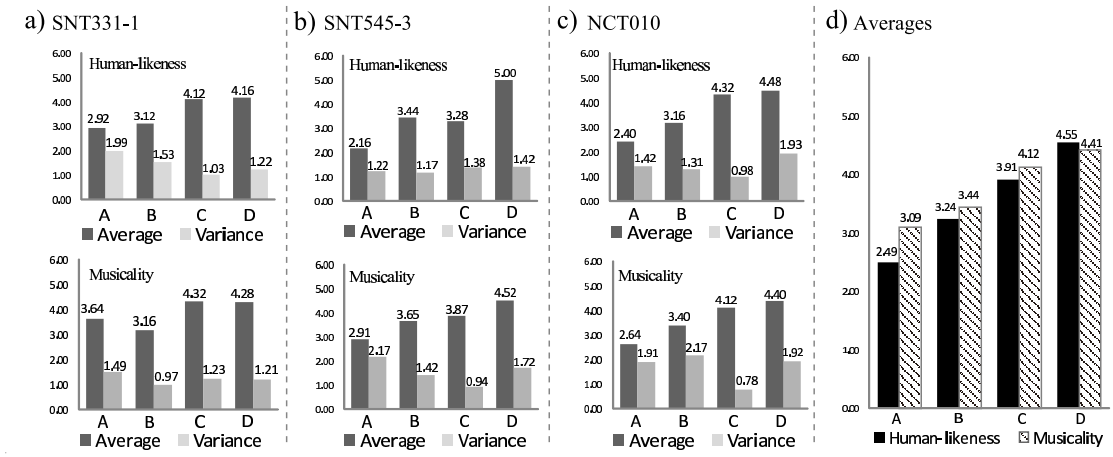


Figure 6. The results of subjective evaluations. A is performances without expression (deadpan). B is performance expression for comparison. C is performance expression with proposed method. D is human performance expression. Note that human performance expression includes local and global expression. a) shows the results for W. A. Mozart, Piano Sonata, KV. 331, 1st Movement and b) shows the results for W. A. Piano Sonata, KV. 545, 3rd Movement. c) shows the results for F. Chopin, Nocturne No. 10, Op. 32, 2nd Movement and d) shows the average human-likeness and musicality of the 3 pieces.

THE INFLUENCE OF REED MAKING ON THE PERFORMANCE AND SOUND QUALITY OF THE OBOE

Carolina Blasco-Yepes
Universidad Politécnica de Valencia, Spain
cablasy@hotmail.com

Blas Payri
Universidad Politécnica de Valencia, Spain
bpayri@har.upv.es

ABSTRACT

An essential part of the oboe technique is the reed-making process, where the raw material is carved and shaped. Different oboe schools define different types of shapes, and argue about their adequacy for a better sound and performance. This paper focuses on the perceptual influence of 3 reed-making types.

We chose 6 reeds representing 3 pairs of each style (French, German, American) and recorded 116 sound samples with two oboists in controlled conditions. N=63 sound stimuli were selected: 9 *diminuendo* long tones, 18 eight-note phrases from which 18 low-pitched and 18 high-pitched tones were extracted. Tones were normalized in pitch and intensity to help listeners focusing on timbre. 40 participants (20 non-oboist musicians and 20 professional oboists) completed a free-grouping task on each of the 4 stimulus sets, grouping sounds by global similarity.

Results show that the most salient production parameters are the attack type and the oboist-oboe. The reed-making shows no significant influence on isolated tones and a marginal influence on complex phrases, and inter-reed differences are more important than inter-reed-making differences. Reed-making is important in performance technique but has no influence on the perceived timbre. Future research will deal with performer proprioception of the reed making

1. INTRODUCTION

There are several hypotheses among musicians and musical schools about the importance of reed-making style in the performance technique and the sound obtained. The experiments presented in this paper try to bring an objective study on what really is perceivable in the sound when selecting different reed types, namely the German, French and American styles [9]. We can decompose roughly the elements that influence the sound in oboe performance in three parts: the instrument body, the performer and the reed. Reeds and their shape are essential in the whole production of sound, and as stated by famous oboe theorist Haynes [8] "c'est l'anche qui donne la vie". In spite of the well known importance of the reed, we must wait until the XIXth century to find the first detailed written instructions on how to make a reed, and we may deduct that previously they were transmitted through oral communication and through handicrafts imitation [3].

From the beginning of the XIXth century on, we have very detailed descriptions on reed making and the problems that may arise when sharpening the reed [1,2,5,11, 15,16,18,19], but only Brod [2] makes a comprehensive study on the different ways to sharpen the reed in the different national schools, comparing the different foreign styles, mainly Italian and German, and comparing them to the French style, and he includes some links between reed making technique and sound quality, describing the German style as *dur* (harsh) and *sourd* (muffled) and proposing the French style as the most convenient for a better oboe performance. In Spain there is also a distinction of sound quality made by theorists [10][11].

We have then an extensive theoretical corpus about reed-making and a tentative description of the sound obtained. The goal of this study is to measure the eventual perceptual differences between reed-making styles. The literature does not provide with a systematic study of the influence of the reed-making the sound production in spite of the repeated theoretical importance attached to this process. Some studies, like Ledet's [9], make a reference to the changes in sound perception but we find hardly any experiment that measures the direct influence of reed making on acoustic measurements or perception changes. A specific study focuses on the effect of the intonation of the crow of the reed on the tone quality of the oboe [13] using basically one reed-making style (the American).

[14] investigated the influence of the reed brand on the acoustic proprieties of the sound with no reference to the reed-making style, and they make a survey with oboists with general questions on the reed and no actual perception of the sound. Their results show that professional oboists respond that they find the reed as more important than the oboe body but they don't show a preference for a specific brand, and the authors provide no significance tests on their results. We find the same lack of significance tests in another experiment [4] that used the responses of 5 oboists on several components of the reed structure: tube diameter, internal gauging, and reed brand.

In order to obtain a significant answer on the perceived and measurable influence of reed-making in sound production, we need therefore to control all the stages from the very process of reed making, reed selection, sound

recording and sound production, and the creation of a perception task that is centered on sound perception and not abstract descriptions of the reed, and following the methodology of timbre perception experiments.

2. REED MAKING PROCESS

Figure 1. Photograph of the 6 reeds that were chosen for recording the sound material of the experiment.

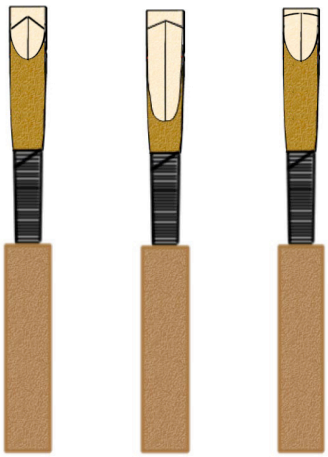


Figure 2. Schematic representation of the 3 styles of reed-making. From left to right: French, American and German. The 3 styles are clearly differentiated by the tip and back areas.

The reed-making process is the essential parameter for this experimental design, and the reeds were made specifically for this research work. We used the 3 main styles defined by Ledet [9]: American, French and German. We tied a total of 25 reeds, using the same manufacturer, staple, internal gouge canes, tube diameter and length. Reed-making was done by hand and when this process was finished, the best reeds with the best material were chosen for the recording. We used a total of 6 reeds, 2 reeds for each type of reed-making. In this experiment, we leave out other aspects of the oboe such as recognition of the different instrument and their schools, the technical aspects of the reeds (type of canes, staples, etc.) or the brand of instruments.

The specific characteristics were as follows:

- Internal gouges: 0’57mm
- Total length (only tied): 74mm
- Total length (cut): 73mm
- Tube diameter: 10'25-10'50mm
- Cane shape: RC12
- Manufacturer of canes: *Le Roseau Chantant*
- Staple: *Chiarugi 47mm 2+*

3. PERFORMANCE RECORDING

3.1 Recorded sounds

Two professional oboists participated in the recording sessions. Both oboists used their own instrument, so that they were used to the idiosyncrasies of their oboe. We made sure that both instruments were of the same type, namely, a Yamaha 831 (professional, semiautomatic), and from the same 4000 series. This provided the maximum similarity in the oboe bodies. Both oboists belong mostly to the German school, and both were trained in Valencia, and were direct or indirect pupils of Lothar Koch. The oboists were instructed to avoid adapting their technique to the reed characteristics. The recording was made in a single session, at the recording studio of the Gandía Campus Radio of the Polytechnic University of Valencia. This room has suitable characteristics for voice and instruments recording. Four AKG C451B microphones were used. These microphones were located one above, one below and two on the sides of the performers, who played sitting and maintaining the same position from the microphones.

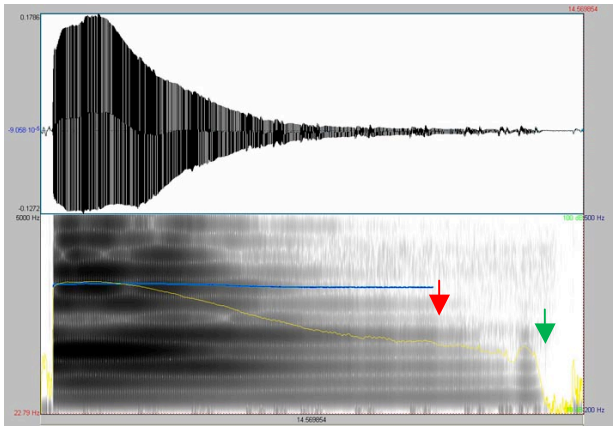


Figure 3. Amplitude/time and sonogram representation of the long isolated tone. The reed arrow points out the end of the stable portion of the sound and the green arrow the end of the perceived sound. The blue line represents F0 (Hz) and the yellow line represents the intensity (dB).

The conditions of temperature (25ºC) and relative humidity (75%) were constant during the 3 hours of the recording session. We recorded 116 sound samples and in this experiment we used the following material:

- Long notes: note G_4, with duration of 12s doing a gradual *diminuendo* (from *fortissimo* to *pianissimo*).
- 8-tone musical phrases: 4 notes G_6 and 4 notes D_4, of 8s. The phrases were played in two ways, first with a tongue attack (i.e. pronouncing the letter "t") and second with a breath attack (without the tongue).
- From each phrase we extracted a low-pitched note (D_6) and a high-pitched note (G_6) with a duration of 1s.

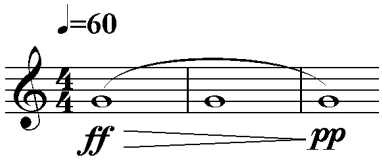


Figure 4. Music notation representation of the *diminuendo* long tone.

The main interest in choosing a long *diminuendo* note is to test the flexibility of the reed. Theoretically, a more flexible reed will allow a larger range of dynamics in the *diminuendo*, although it may induce a progressive drop in pitch. Figure 3 displays the sonogram of a *diminuendo* tone with an indication of the stable and unstable part of the sound. Figure 4 shows the musical notation the musicians had to play.

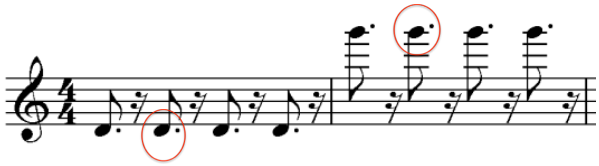


Figure 5. Music notation representation of the eight-tone musical phrase. The notes that were used in the short-tone experiment are circled out.

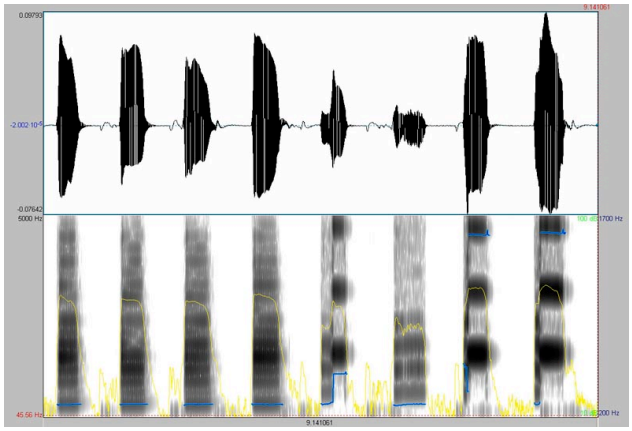


Figure 6. Amplitude/time and sonogram representation of an eight-tone musical phrase. The blue line represents F0 (Hz) and the yellow line represents the intensity (dB).

The 8-note phrase, displayed in figure 5 with musical notation, was used to test the response of the reeds to the different ways to attack a sound. This tests the rigidity of the reed, as some reeds respond better to the air pressure during an attack and some may over-vibrate and play a harmonic tone or fail to sound properly. The tongue attack is sharper and produces an instant increase in pressure. In figure 6 we can see an 8-tone sequence using a tongue attack where some notes fail to sound at the right pitch and intensity.

From the phrase we isolated a low pitched tone and a high pitched tone, selecting preferably the second note of the group as illustrated in figure 5, and if the second note was failed, using the third, so that we had a note with a stabilized sound production without the possible irregularities of the initial or final note of the group. The isolated tones were used to study the timbre with the different reeds and attacks, and the long phrases were used to judge the general stability of the reed with the given attack. The notes that were played use the lower register and higher register of the oboe, with an abrupt change within the phrase that tests the general equilibrium of the reed.

3.2 Acoustic measurements and sound normalization

We did a preliminary acoustic study on the same material and this shows that there are significant differences in pitch according to the reeds and styles of reed-making. Particularly, the German school differed from the American and French.

This experience focuses on the timbral differences between different reeds and reed-making styles, and the most accepted framework for timbre comparison is to use sounds of equal pitch, intensity and duration. Thus we normalized each sound in intensity and pitch, using the theoretical frequency of the note (G_4, D_4, G_6). Even if pitch differences were small, when comparing sounds any pitch change becomes very salient and overcomes other features. Pitch normalization was made by decelerating or accelerating the frequency, therefore we introduced no timbre change, and as the difference in pitch was small (less than a semitone) the speed change has not caused any significant alteration of duration.

4. TIMBRE PERCEPTION EXPERIENCE

4.1 Holistic perception task

The main goal of our experiment is to analyze the effects of reed making on sound timbre. Therefore we have used holistic listening as it is the most accepted perceptual task in timbral studies as described by Grey or McAdams [6,7,12]: in this kind of task, listeners group the sounds according to the global perceived similarity. The timbre research experiments mostly compare inter-instrument differences, but this method is also perfectly suitable to detect timbral differences within the sounds of an instrument. As mentioned before, we normalize the sounds in pitch and intensity, as well as tone duration, as the most accepted working definition of timbre is what differentiates tones of equal pitch, intensity and duration. With the groupings made by each listener, a matrix is made, with 0 when two sounds do not belong to the same group and 1 when they do. The individual matrices have all the properties of a distance, and they are aggregated into a matrix by adding the different values of each listener. The resulting matrix fits again the properties of a distance and can

be analyzed with multidimensional scaling techniques to discover the main axes of the perceptual space of the set of sounds.

We insist in the fact that by using a free grouping task, with no previous information on the sounds and no directions on how to classify them, we hope that the listeners will focus on the actual perception of the timbre, and will not try to use their knowledge of reed-making. The goal is to understand if there is any timbre variation associated with reed making, and furthermore, to understand the relative importance of each factor in the global timbre perception, such as differences in the performer, the instrument or the idiosyncrasies of each reed.

4.2 Task

We created four sets of sounds, and each set was presented independently to the listeners. Once the listener had completed the grouping of a given set, the next set was loaded. The sets increased in stimulus complexity and diversity. To establish the order of presentation, we took in account the fact that we had 8-note phrases, and isolated notes from these phrases. As the goal was to force a reduced listening as described by Schaeffer [17], the isolated tones were presented first and the phrase last, presuming that if listeners heard the phrase first, they would try to match the isolated note grouping to what they had done with the complex samples. The *diminuendo* set was the simplest as it had less items which varied less, as no attack differences were present. The sets were presented in this order:

- Stimuli set 1: 9 *diminuendo* long tone (G_4).
- Stimuli set 2: 18 short-tone low-pitched notes (D_4), 9 with a tongue-attack and 9 without.
- Stimuli set 3: 18 short-tone high-pitched note (G_6), 9 with a tongue-attack and 9 without.
- Stimuli set 4: 18 phrases with 4 low-pitched note (D_4) and 4 high-pitched notes (G_6), 9 with a tongue-attack and 9 without.

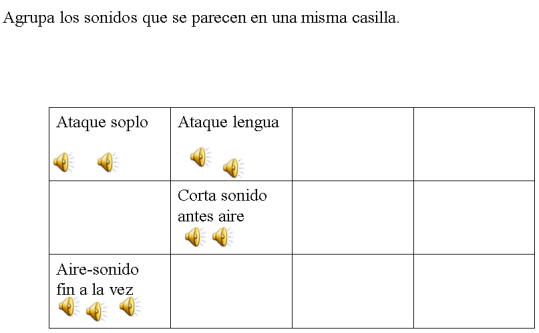


Figure 7. Screenshot of the interactive tool developed for free-grouping sound perception.

Listeners used an interactive interface to listen the sounds and drag the corresponding icons as shown in figure 7. The interface presented the listeners with a set of sounds (associated with an icon that they could freely move and listen) and they had to group them according to their global similarity (holistic listening). In addition, the listeners could write in the boxes a description of the group of sounds and its differentiating features (free verbalization). The listening task was done individually, with headphones and had an approximate duration of 20 minutes.

4.3 Participants

40 participants (20 non-oboist musicians and 20 professional oboists) completed a free-grouping task.

5. RESULTS

5.1 Data

Each classification of a set of sounds corresponds to a matrix which fits all the properties of a distance and it is calculated by adding 1 every time a pair of sounds have been grouped apart.

5.2 Response coherence

We tested the inter-rater response coherence by measuring Cronbach's Alpha, as displayed in figure 8. We can see that the group of oboists has a higher inter-rater coherence, probably due to the fact that oboists are more used to listen analytically the sounds of the oboe and may have common criteria acquired in the learning process.

There is also a clear difference between the *diminuendi* tones and the rest: the *diminuendi* did not have differences in the attack where as in the rest of sounds sets we had 2 types of attack. This has reduced the production differences, and therefore the listeners did not have clear criteria to group the sounds. We will see (figures 10 & 11) that the reed-making differences have not been perceptually very salient. When in a set of items we reduce the factors that may differentiate them, the raters should focus on the remaining factors if they are perceptually relevant. In this case, the *diminuendi* had the reed-making and the performer as production differences, and instead of focusing on reed-making sound differences, they just did not have a consistent grouping, so we can hypothesize that reed-making is not perceptually very relevant for the sets of sounds we have created. Differences between sound sets in the behavioral results may measure true differences in the perceptual qualities of the tones, but we can also consider that the effect of increased practice with the sorting task has derived in a increased agreement between listeners, therefore increasing the Cronbach's Alpha value.

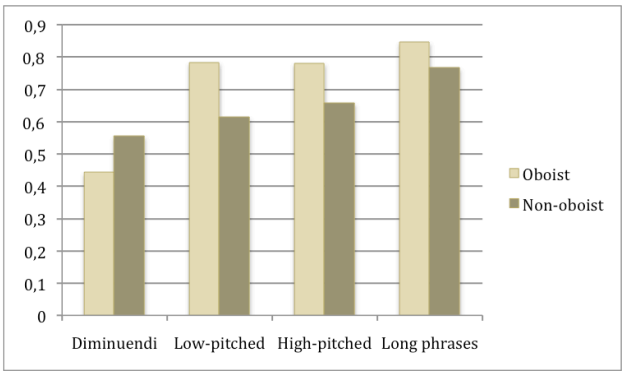


Figure 8. Cronbach's Alpha values for the responses of the two groups of listeners in each set of sounds.

5.3 Multidimensional scaling

We performed a multidimensional scaling on each of the response matrices, using a bi-dimensional model (as the tri-dimensional model did not explain much more variability) to detect the main criteria used by the listeners. The variability explained by the scaling was RSQ= 65% for set 1, RSQ=79% for set 2, RSQ=76% for set 3 and RSQ= 91 % for set 4. As we can see, the variability explained is low except for set 4, and only for set 4 the dimensions obtained could be explained, as shown in figure 9. On the figure we circled out the groups corresponding to the MDS analysis: horizontally (dimension 1) two groups can be divided according to the attack type: breath versus tongue and we have circled out with a dashed line the sounds of breath attack; vertically (dimension 2) also two groups appear according to the performer, we have displayed the sounds of the 2d oboist with a solid line.

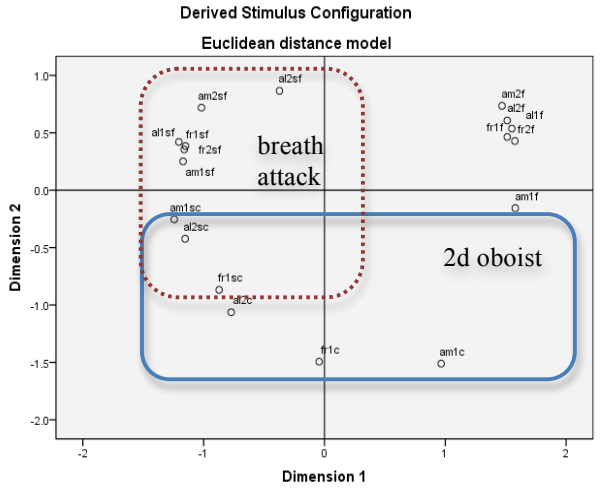


Figure 9. Multidimensional scaling of the stimuli set 4. The first 2 letters are related to the style of lowering: letter S shows a breath-attack. Last letter shows the performer.

5.4 The relative influence of the factors

Tests (Pearson) χ^2				
Stimuli	Participants	χ^2	df	Asymp. Sig. (2-sided)
Diminu-endi	Non-oboist	13.0**	1	**.000**
	Oboist	11.7**	1	**.001**
Low-pitched	Non-oboist	2.15	1	.142
	Oboist	5.27*	1	**.022**
High-pitched	Non-oboist	7.78**	1	**.005**
	Oboist	.403	1	.525
Long phrases	Non-oboist	11.1**	1	**.001**
	Oboist	48.3**	1	**.000**

* significant to the 0.05, ** significant to the 0.01

Table 1. Pearson χ^2 test comparing the responses differences for the factor "performer" for each set of stimuli.

To determine if the different styles of reeds affect the perception of listeners, we have done a test of χ^2 dividing the participants into two groups: professional musicians and oboists. We have compared if there were significant differences in the groupings of listeners using the following factors: the oboist (2 oboists), attack (2 types of attack, with the exception of *diminuendi* stimuli which present only one type of attack), reed-making style (3 styles) and reeds (6 reeds: variability inter-reed and intra-school).

Tests (Pearson) χ^2				
Stimuli	Participants	χ^2	df	Asymp. Sig. (2-sided)
Diminu-endi	Non-oboist	.140	1	.709
	Oboist	3.91*	1	**.048**
Low-pitched	Non-oboist	.237	1	.626
	Oboist	.501	1	.479
High-pitched	Non-oboist	1.43	1	.231
	Oboist	1.79	1	.181
Long phrases	Non-oboist	5.44*	1	**.020**
	Oboist	.916	1	.338

* significant to the 0.05, ** significant to the 0.01

Table 2. Pearson χ^2 test comparing the responses differences for the factor "reed-making style" for each set of stimuli.

In all the stimuli, the results show that the elements that affect most the perception of the participants are the attack type (table 2) and the performers (table 1), and these differences are significant for both professional musicians and oboists listeners. The performer is not a significant parameter only for short-tone high-pitched notes in the oboists listener group.

As can be seen in table 2, the factor with the least significant differences is the reed-making style; these differences appear only for sentences of 8 notes in the group of listeners who are not oboists (p=,020). In *diminuendi* cases we can observe a significant difference for oboist listeners (p=,048).

Tests (Pearson) χ^2				
Stimuli	Participants	χ^2	df	Asymp. Sig. (2-sided)
Diminu-endi	Non-oboist	.566	1	.452
	Oboist	3.37	1	.066
Low-pitched	Non-oboist	.745	1	.388
	Oboist	9.81**	1	**.002**
High-pitched	Non-oboist	3.99*	1	**.046**
	Oboist	2.51	1	.113
Long phrases	Non-oboist	6.22*	1	**.013**
	Oboist	18.1**	1	**.000**

* significant to the 0.05, ** significant to the 0.01

Table 3. Pearson χ^2 test comparing the responses differences for the factor "reed" for each set of stimuli.

By contrast, we can see in table 3 that the differences between individual reeds (independently of the making style) are relevant for both oboists and non-oboists for the

low-pitched tones and even more for the complete phrases. In figure 10 we summarize the results of the different χ^2 tests we have performed. As we can see, the attack type is by far the most relevant factor when it is present (it is not present in the *diminuendi* set). The attack type has a significant influence in every case. To have a better comparison for the remaining factors, we eliminated the attack as shown in figure 11.

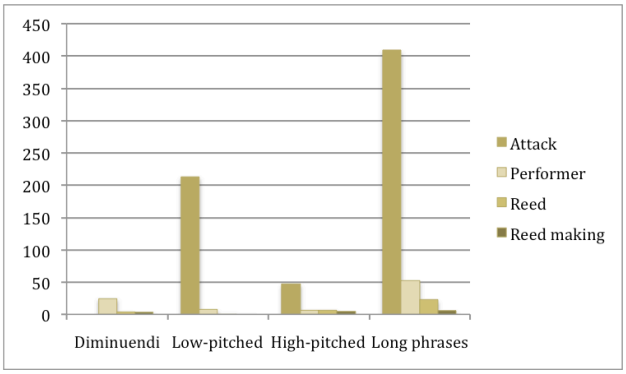


Figure 10. Graphical display of the χ^2 values for the four factors under study.

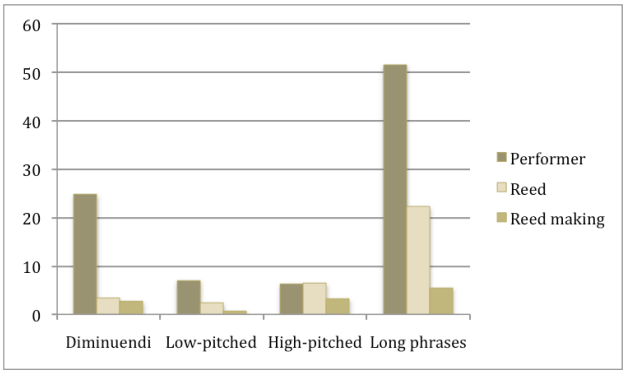


Figure 11. Graphical display of the χ^2 values for the three factors excluding the attack type.

Tables 4, 5, 6 and 7 describe the results for the Pearson Chi-square tests using the responses from all the participants and using as parameter the four factors under study. These results are graphically summarized in figures 10 and 11.

	Tests (Pearson) χ^2		
Diminuendi	χ^2	df	Asymp. Sig. (2-sided)
Attack	-	-	-
Performer	24.738**	1	**.000**
Reed	3.31	1	.069
Reed making	2.71	1	.100

* significant to the 0.05, ** significant to the 0.01

We see in figure 11 that the performer is the most influential factor after the attack, and it is also significant for all the sets. For the *diminuendo* set the importance of the performance is higher than for the other isolated note sets (low and high pitched notes), which can have two causes: the *diminuendo* notes are longer and therefore listeners may distinguish better performance differences, and also, the *diminuendo* notes were performed with the same type of attack; when the most salient feature (attack) is suppressed listeners may focus their attention to the remaining performance differences. As can be seen in the tables, the inter-reed differences are more salient than the inter-reed-making style differences. This means that the specific characteristics of the reed overcome the influences of the reed-making. Furthermore, the reed-making only becomes significant with the 8-tone phrases, where listeners can focus on performing technique (missing notes, performing regularity in the different notes) rather than on mere timbral differences.

Table 4. Pearson χ^2 test results for the *diminuendo* set.

	Tests (Pearson) χ^2		
Low-pitched	χ^2	df	Asymp. Sig. (2-sided)
Attack	213.156**	1	**.000**
Performer	7.01**	1	**.008**
Reed	2.393	1	.122
Reed making	.708	1	.400

* significant to the 0.05, ** significant to the 0.01

Table 5. Pearson χ^2 test for the Low-pitched set.

	Tests (Pearson) χ^2		
High-pitched	χ^2	df	Asymp. Sig. (2-sided)
Attack	47.099**	1	**.000**
Performer	6.211*	1	**.013**
Reed	6.467*	1	**.011**
Reed making	3.181	1	.075

* significant to the 0.05, ** significant to the 0.01

Table 6. Pearson χ^2 test for the High-pitched set.

	Tests (Pearson) χ^2		
Long phrases	χ^2	df	Asymp. Sig. (2-sided)
Attack	408.233**	1	**.000**
Performer	51.552**	1	**.000**
Reed	22.344**	1	**.000**
Reed making	5.511*	1	**.018**

* significant to the 0.05, ** significant to the 0.01

Table 7. Pearson χ^2 test for the Long phrases set .

6. CONCLUSIONS

The main result of our experiment is that reed-making style has a minimal impact on the global shaping of timbre. Performance-related features like the type of attack or the technique of the performer become more salient

when they are present and are the only features that consistently influence perception. Multidimensional scaling shows that there is no common strategy, and in particular no strategy associated with reed making, when listeners deal with isolated tones; whereas there are some significant results when listening to 8-note phrases due to attack and instrument differences. Bearing in mind that the isolated tones (high or low-pitched) come from the phrases, and even if it is the same original sound material, the length and added complexity of the phrases bring out different perceptive properties. Long stimuli (isolated *diminuendo* tones and 8-note phrases) bring a more significant distinction between performers, as indicated in table 1. In no case we can detect that reed-making style is a dimension of sound similarity. We can conclude therefore that there is no consistent timbre variation associated with reed making in our material.

First, we point out that the difference between isolated tones and phrases is interesting, as many studies on musical timbre or performance do not compare the results on these two kinds of material. The fact that longer and more complex samples bring coherent classifications can mean that listeners focus their attention on technical performing differences: both performers reported that some sound samples were easier to achieve with one sort of reed than with others, and, for instance, in some cases we have a longer delay in the onset of the high-pitched notes that come after the low-pitched notes, or some notes are not stable more frequently with one type of reed than others. Reed making may therefore influence the technical ability to achieve some technical requirements and will influence the global performance but not particularly the timbre, which is the focus of this paper. The fact that the sounds were normalized in pitch and intensity may also be a factor in the lack of consistent classification strategies, as these two factors are always most salient for trained and untrained listeners; a previous study showed that there were some significant differences between reeds in tuning and other acoustic parameters.

Second, we have detected an important gap between the sound perception and the comments provided by the performers: the two oboists that recorded the sound samples reported differences between the reeds in general and between reed-making styles in particular. Although we did not realize a systematic study of the performers perception, we are inclined to believe that there is indeed a strong mechanical and haptic difference as reed making is consistently reported as an essential part in the oboe functioning, by oboe practitioners and learning manuals. Our future research is going to study systematically the proprioception of the oboe performer, comparing it with the sound perception. For that matter, we are going to use a large set of reeds made using mechanical sharpeners that yield always the same shape. The oboists will play a certain number of exercises that will stress technical qualities of the reed and the oboe, and for each exercise, each oboist will report the proprioception about the elasticity, resistance, ease of use for intonation or for the different types of attack, etc. The resulting exercises will be recorded and the sounds will be used in further tests of sound perception to detect if there are some features of reed making that are sensitive to the performer but not to the audience.

7. REFERENCES

[1] A. M. R. Barret: *A Complete Method for Oboe*, Jullien and Co., London, 1850.

[2] H. Brod: *Méthode pour le hautbois (Vol 1 & 2)*, Schonenberger, Paris, 1830.

[3] G. Burgess & B. Haynes: *The Oboe*, Yale University Press, New Haven & London, 2004.

[4] O-H. Dahl: *Better Oboe Reeds*, Mediefabrikkens ApS, Copenhagen, 2001.

[5] J. F. Garnier: *Méthode raisonnée pour le hautbois*, Pleyel, Paris, 1800.

[6] J. M. Grey: "Multidimensional perceptual scaling of musical timbre," *The Journal of the Acoustical Society of America*, Vol. 61, No. 5, pp. 1270-1277, 1977.

[7] J. M. Grey: "Timbre discrimination in musical patterns," *The Journal of the Acoustical Society of America*, Vol. 64, No. 2, pp- 467-472, 1978.

[8] B. Haynes: *The Eloquent Oboe. A History of the Hautboy from 1640 to 1760*, Oxford University Press, New York, 2001.

[9] D. Ledet: *Oboe Reed and Styles. Theory and Practice (2nd ed.)*, Indiana University Press, Bloomington, 2008.

[10] V. Llimerà: El Método de Oboe de Enrique Marzo y Feo (1870), PhD Thesis, Universidad de Valencia, Spain, 2006.

[11] E. Marzo: *Método de Oboé progresivo y complete con nociones de Corno Inglés*, Antonio Romero, Madrid, 1870.

[12] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, & J. Krimphoff: "Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes," *Psychological Research*, Vol. 58, No 3, 177-192, 1995.

[13] J. C. Prodan: "The Effect of the Intonation of the Crow of the Reed on the Tone Quality of the Oboe," *Journal of the International Double Reed Society,* Vol. 5, 1977.

[14] J. Romero, J. Alba & J. Ramis: "Estudio preliminar del comportamiento de cañas de oboe," *Proceeding of the 37ª Congreso Nacional de Acústica TecniAcustica*. Universidad Politécnica de Valencia, Spain, 2006.

[15] C. Salviani: *Metodo completo per oboe cotento nozioni preliminari*, Lucca, Milan, 1848.

[16] J. Sellner: *Méthode pour le hautbois*, Richault, Paris, 1830.

[17] P. Schaeffer: *Traité des objets musicaux, 2nd edn*., Editions du Seuil, Paris, France, 1977.

[18] L. A. Veny: *Méthode abrégée pour le hautbois*, Pleyel et Cie, Paris, 1828.

[19] A. Vogt: *Méthode pour le hautbois "du Celèbre Vogt offerte par Mr. Bruyant son Élève à la Bibliothèque du Conservatoire 1872"*, National Library of France Ms. Ci 50, Paris, 1816-1824.

KETTLE: A REAL-TIME MODEL FOR ORCHESTRAL TIMPANI

Panagiotis Papiotis
Music Technology Group,
Universitat Pompeu Fabra
panos.papiotis@gmail.com

Georgios Papaioannou
Department of Informatics,
Athens University of Economics and Business
gepap@aueb.gr

ABSTRACT

The orchestral timpani are a key component in western classical music, although their weight, size, and fragility make their transportation very difficult. Current commercial software synthesizers for the Orchestral Timpani are primarily sample-based and work with a MIDI keyboard, giving the user control over the note amplitude and pitch. This approach implements a virtual five-piece set of orchestral timpani, which is controlled using a pressure-sensitive graphics tablet. A brief analysis of the mechanics and playing techniques of the Timpani is presented, followed by their approximation by this model's control scheme and sound engine. Thereon, the details of the model's implementation are explained, and finally the results of the model are presented along with conclusions on the subject.

1. INTRODUCTION

The Timpani (also known as *kettledrums*) are a type of drum and, as such, belong to the musical family of percussion instruments; they consist of a thin membrane or *head* stretched over a large bowl traditionally made of copper. Unlike most drums, the timpani are capable of producing a strong pitch sensation when struck, and can be tuned via a mechanism that adjusts the tension of the membrane over the bowl. The most commonly used type of timpani today is the *pedal timpani*, in which the player controls a pedal with his/her foot which either increases or decreases the tension of the membrane, thus altering the drum's pitch. Older variations of timpani (e.g. the timpani used in the baroque period) are tuned by adjusting the tuning bolts individually.

Timpani are primarily used in sets of two or more, according to the music piece's requirements and, of course, the number of timpani available (it is not unusual for percussionists to play with one less drum than the piece requires, due to limitations). The largest of the commonly used timpani sets consists of five timpani, each with a diameter of roughly 32, 29, 26, 23 and 20 inches respectively.

Following the advances in software synthesizers, several commercially available products included sample-based models for the Orchestral Timpani as part of a larger Virtual Instrument Suite (examples include Edirol's *HQ-Orchestral*, IK Multimedia's *Miroslav Philharmonik* and Eastwest/Quantum Leap's *Advanced Orchestra Set*). These models, while useful for composers that wish to add to their compositions a few sporadic notes or pre-recorded timpani rolls (fast consecutive hits that create a rumbling sound), are designed to be controlled by a MIDI keyboard and thus do not take advantage of the instrument's continuous pitch range nor its vibrational mode-dependent timbre.

Our approach implements a real-time system that aims for two goals: to give the user control over every important factor that shapes the final sound of the instrument, while keeping the control scheme as simple and intuitive to the user as possible. A third and equally important goal was for the system to be easily available to potential users without the need for advanced control interfaces or high-performance computer systems.

The remainder of this paper is organized as follows: In Section 2, previous work on the subject is presented. Section 3 provides a brief overview of the instrument's unique sound mechanics, as well as the specific techniques employed by professional timpanists in order to achieve different sounds. Section 4 presents the design and architecture of our model. Section 5 contains the final model implementation details. Section 6 presents our results. Finally, Section 7 contains our conclusions as well as recommendations for future work.

2. PREVIOUS WORK

An important amount of research has gone into modeling the sound of percussion instruments and their abstractions (such as ideal membranes, plates etc.). One of the most promising and mature fields is *Physical Modeling* [1], or the simulation of the sound generation mechanism for a particular instrument. The clear advantage of this approach is that, through Physical Modeling, most parameters that partake in the creation of the sound are adjustable, thus making meaningful interaction with the model very intuitive for the user.

In this field, a very prominent approach is the use of the *Digital Waveguide Mesh* [2], in which a membrane is simulated by a set of one-dimensional digital waveguides, very much like a tennis racket's strings simulate an elastic membrane. Two-dimensional and three-dimensional realizations efficiently simulate membranes and acoustic spaces, while a significant improvement over this ap-

proach is *frequency warping* [3] in an interpolated three-dimensional mesh. Besides these cases, recent work on modeling rigid bars and plates [4] is also promising. An especially interesting combination of physical modeling for percussion instruments and multi-dimensional user input can be found in [5]. In this approach, a two-dimensional force matrix is used to both excite and damp a waveguide mesh, attempting to simulate the expressive qualities of hand drumming.

Another approach for two-dimensional membranes is presented in [6], making use of the so-called *Functional Transfer Methods* [7] - utilizing an analytical form of the modal parameters for a multi-dimensional differential system such as a membrane. There are more approaches to physically modeling nonlinear circular membranes [8], as well as two-membrane air coupling systems with strings such as a snare drum [9]. However, these approaches are relatively far from the specific physiology of the Timpani, and will not be discussed further.

There exist two complete numerical models specifically targeting the Timpani [10,11], and experimental results are promising. In [11], the model delves into the complexities of the Timpani as a physical system with great detail, taking into account the motion of the mallet and its nonlinear interaction with the membrane, the transverse displacement of the membrane, and the sound pressure inside and outside the enclosed cavity of the resonator bowl. Unfortunately, the aforementioned complexity of these models makes them unsuitable for a real-time system.

Finally, a very interesting and similar to this work project can be seen in [12], where a hardware sample-based model attempts to simulate the sound and control scheme of the kettledrum. This project is based on a custom-created drum pad with piezoelectric transducers, and sends input data from an electric circuit board via the RS-232 protocol to a PC where the audio output is calculated and played back. This project features a single kettledrum, one type of mallets, and uses an analog foot pedal to tune the drum.

3. TIMPANI SOUND PROPERTIES AND PLAYING TECHNIQUES

In order to provide a context for the subject at hand, key principles and terms related to the sound and playing conventions of the Timpani will be briefly presented. This is essential as this will be the base on which our model is designed.

3.1 Sound properties of the Timpani

As it was mentioned in the introduction, what distinguishes timpani from other drum-type instruments is the fact that they are able of producing a strong pitch sensation. The pitch is controlled using a tuning mechanism, and the tonal range of each drum is usually a perfect fifth (depending on the condition of the drum as well as its size). On certain types of Timpani, this range can be expanded to a full octave. Since the pitch is altered in a continuous manner by simply modifying the tension of the membrane, the Timpani are capable of producing any possible pitch within their tonal range, including micro-tonal intervals as well as "glissandi" or *sliding notes*.

Another important factor that sets Timpani apart from drums with a smaller skin surface, is the fact that they are highly sensitive to the distance of each hit from the center. Since circular membranes are two-dimensional, they can vibrate in many modes simultaneously and most of these modes are not harmonic; that is the frequencies of higher modes are not *n*f0, n=1, 2, …, k* (where f0 is the fundamental frequency and k the number of harmonics) as is the case with the harmonic series. Moreover, vibration in two dimensions means that there are two vibrational modes simultaneously and therefore two sets of nodal points; nodal circles and nodal diameters [13].

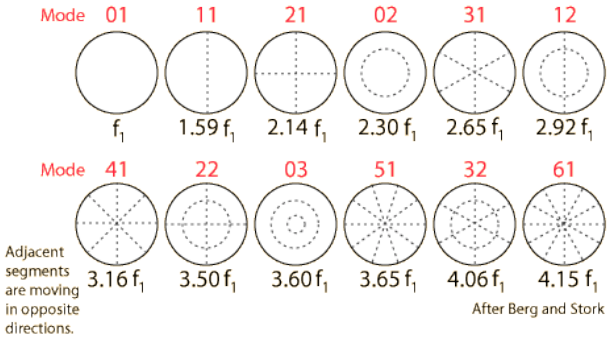


Figure 1. Vibrational Modes of a circular membrane.

When the (01) mode is excited, there are zero nodal diameters and one nodal circle on the boundaries of the membrane (see Fig.1). When the (11) mode is excited, there is one nodal diameter and one nodal circle, et cetera. However, not all of these modes produce a musically meaningful result; for this reason, there is a set of *preferred modes* [13], which timpanists and composers alike are very familiar with and exploit to achieve a different timbre that matches the musical occasion.

3.2 Playing techniques & conventions

As with most percussion instruments, the choice of mallets in Timpani performances is very important. There are several types of mallets which vary on the hardness of the mallet head and the material it is made of, as well as the elasticity of the mallet's shaft. Almost every musical piece instructs the player on the type of mallets to use, ranging from soft felt mallets, flannel mallets, harder mallets used also for marimbaphones, and mallets with wooden heads. Each type changes not only the attack of each note, but also the timbre that is produced.

Most musical pieces written for/including timpani also demand that the timpanist tunes the drums as he/she plays. Almost all advanced pieces change the set's tuning during the piece, and many require that the timpanist changes the tuning of a drum while it is still resonating.

This sliding note technique is called *glissando*, and is commonly used in conjunction with timpani rolls.

The concept of a drum roll is central to all percussion instruments, and it consists of fast consecutive strikes on the drum's surface that alternate between the player's left and right hand. This is especially important for the timpani, as it creates a rumbling, drone-like sound which is pivotal all timpani pieces.

Finally, a known technique in music for timpani is membrane *damping*. A timpanist often touches the membrane in order to mute that drum; another technique is the use of *mutes*, which resemble small pillows and are placed on the membrane throughout the musical piece while the timpanist plays, in order to reduce the vibration and "dry out" the sound produced.

4. DESIGN OF THE MODEL

In order to satisfy the model's requirements in terms of sound synthesis and user interaction scheme, these two attributes needed to take full advantage of each other. Therefore, in the following section we present our choice for the model's sound engine and control scheme, as well as the final design of the system.

4.1 Sound synthesis

We considered two different approaches for the sound synthesis engine: Physical Modeling, and Sample-based synthesis.

Judging by the quality of the existing physical and numerical models for timpani presented in section 2, our initial idea was to implement our model based on a physical modeling synthesis algorithm such as the digital waveguide mesh, coupled with a model for the drum's resonator bowl. Considering recent advances in the field, it is easy to deduce that an extremely realistic physical model would produce the best results; on the other hand, physical models include numerous mathematical calculations which, in a three-dimensional system such as the kettledrum, would not only require significant processing power but also potentially introduce numerical errors in order to compute the output in real time.

The second approach would be a sample-based model; such a system is implemented in the commercially available timpani models, and could be partially connected to the pathologies shared by them. However, one cannot separate the interface from the virtual instrument: MIDI keyboards only allow for discretized pitch and note amplitude based on the velocity and number of the key pressed. Such a system is effective for instruments with keys (such as pianos or organs), or even some string and wind instruments. If an appropriate control interface provides more input dimensions, the manner in which samples are interpolated to produce the synthesized sound can produce very convincing results [1]. Moreover, sample-based synthesis is far less computationally intensive than a physical model, which is very important in a real-time application. For these reasons, we decided to implement a sample-based model, which is controlled by an appropriately multi-dimensional interface.

4.2 Control scheme

As described in section 3, the most important parameters that shape the sound of the Timpani are the *hit coordinates*, the *type of mallets* used, the *tuning* of each drum, the use of *damping/mutes,* and of course the *strength* of the hit.

From the above parameters it is obvious that the interface used to control our model must possess at least two important qualities; provide two-dimensional coordinates for the position of the mallet's strike, and be pressure-sensitive. All other parameters (type of mallets, tuning, and mutes) can be selected using the existing controllers in a computer, such as the mouse and/or keyboard.

There are several interfaces that provide two-dimensional, pressure sensitive input. However, one of our goals was to design a system that would be available to the majority of potential users; thus, we had to exclude expensive controllers such as the hit coordinate-sensitive Mandala drum pad [14] or touch screens, as well as custom-designed interfaces such as pads equipped with multiple piezoelectric transducers.

Another alternative would be a webcam-based approach; unfortunately, ordinary webcams have a relatively low frame rate, and are prone to errors caused by distractions within the capture area.

As our system required a widely available but at the same time relatively precise interface, our choice was to use a digitizer tablet. Although tablets do not provide tactile feedback (such as the previous solutions), they are accessible due to their low starting price (~50 Euros), and provide high capture precision (albeit with a restricted surface area, especially for cheap models). Moreover, most digitizer tablets also feature scroll wheels and programmable buttons; these extra controls can be used to change the tuning of the timpani as well as the mallet type.

4.3 The model

Having decided to use a sample-based synthesis algorithm, we needed to determine the number and nature of the samples that would comprise the soundbank for our model.

4.3.1 Hit coordinates

In timpani sheet music, there are predominantly four different written notes to signify the coordinate of the hit (see Fig. 2).

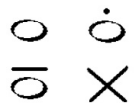


Figure 2. Sheet notation for hit coordinates.

From top to bottom and left to right, these symbols specify the hit position as *normal* (i.e. at ¾r from the drum's center, where *r* equals the radius of the drum), *staccato* or close to the drum's center, *legato* or close to the drum's rim, and the final symbol signifies a hit either on the cen-

ter or on the rim of the drum's head, further explained using footnotes on the sheet music. Thus, five different hit positions were recorded in the soundbank.

4.3.2 Dynamics

In order to have a dynamically varied model, it is not sufficient to play back the samples at different levels of amplitude. A kettledrum's sound differs greatly when it is struck softly rather than strongly, as the harmonics and partials present in the attack and release parts of the note evolve differently. Therefore, it was pivotal to the model's realistic nature that several dynamic levels were recorded.

In classical music, dynamics range from *ppp* (pianissisimo, very very soft) to *fff* (fortissisimo, very very loud). Of all those dynamic values, the most commonly met are *pianissimo*, *piano*, *mezzo piano, mezzo forte, forte, and fortissimo*. Dynamics outside this range (as well as the *mezzo piano* dynamic) are meaningful musically, but can be achieved by adjusting the level of a recorded sample since their distance from the next closest value is small. In our model, the samples were recorded in five different dynamic values (pp, p, mf, f, ff).

4.3.3 Types of mallets

There are many different types of mallets that can be used in order to create a unique sound. Often timpanists freely choose the type of mallet to achieve a specific sound, even if it contradicts the piece's instructions. Overall, timpani mallets fall into three categories: *soft* mallets, with heads typically made from wool or flannel; *hard* mallets with wooden heads wrapped in thick thread resembling those used for chromatic percussion such as the Marimba or the Vibraphone; and finally *wooden* mallets, or mallets with wooden heads that are wrapped in leather.

In order to control the size of the soundbank, the samples were recorded with three different pairs of mallets, with each pair being the most commonly used in its category: soft wool mallets, Marimba mallets with thread-wrapped heads and bare wooden mallets.

4.3.4 Tuning range

Western music theory divides an octave in 12 intervals, the *chromatic scale*. Of these 12 intervals, only eight fit in the tonal range of a typical kettledrum. However, to record all 8 notes would not only result in an exceedingly large soundbank, but would also deprive the model of the ability to *slide* between notes, achieving microtonal intervals and performing glissandi. Thus, an efficient real-time algorithm had to be created so that each drum could achieve any possible tuning between its lowest and highest possible note.

The most obvious way to achieve such a result would be to simply pitch-shift a middle note, in order to emulate higher and lower pitches. However, it was evident early in out experiments that such a solution would neglect the difference between the harmonic series present in two notes with different pitch (see Fig.3).

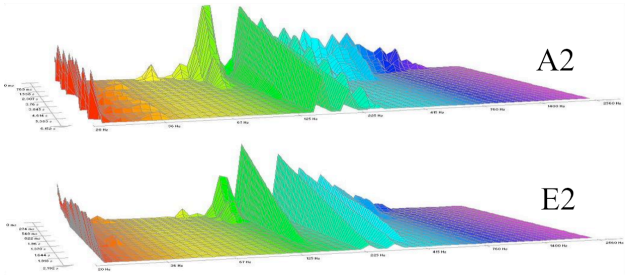


Figure 3. A2 and E2 played on a 29'' kettledrum.

The most important segments of a Timpani note are respectively its *attack* and *decay*, as is the case with most instruments. Figure 4 shows the spectrogram of a C2 note played on a 29'' kettledrum:

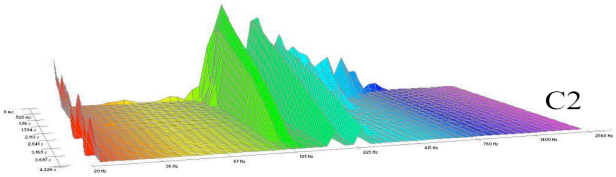


Figure 4. C2 played on a 29'' kettledrum.

The difference between Figure 3 and Figure 4 is evident: to pitch-shift either of these notes in order to match the other would distort the original timbre of the note we are trying to get. Figure 5 shows the spectrograms of two different C2 notes, produced by pitch-shifting an A2 and an E2 respectively:

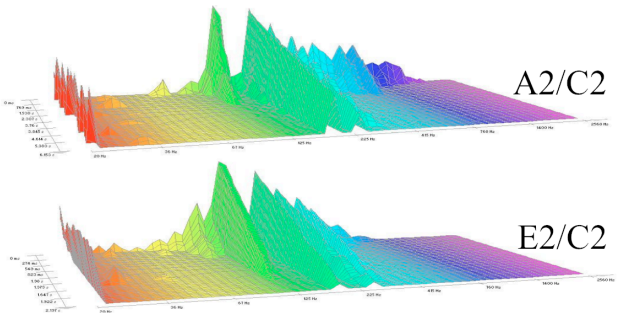


Figure 5. A2 and E2 pitch-shifted to reach C2.

It can be observed in Figure 5 that each shifted note has different similarities to the actual C2 note. Namely, the *attack* part of the pitched-down E2 note is a good approximation of the C2 note's attack, whereas the *decay* part of the pitched-up A2 note matches with the decay of the natural C note. Of course, this can be observed better acoustically, but a common element that is present both acoustically and visually is that, while the pitched-down note provides a more precise representation of the harmonics contained in the actual note, the pitched-up note retains a better approximation of the duration and time evolution of the original note (since the duration of the

pitched-down E2 note is a mere 2.2 seconds, while the duration of the pitched-up A2 note is 6.2 seconds).

Taking advantage of this knowledge, we implemented a crossfading algorithm that gradually combines the two pitch-shifted notes to create the final note. This way, we managed to both restrict the size of our soundbank, and to achieve all possible tunings within each drum's tonal range.

Pseudocode for this crossfading algorithm can be seen in Figure 6. In essence, our engine utilizes two notes, each one at the boundary of the kettledrum's tonal range; these notes are aptly named the *high* and *low* note, respectively. Using time-domain pitch shifting, the *high* note is shifted downwards and the *low* note is shifted upwards to match the desired pitch.

```
for the first 1/10 of the high note
      combined note = 1*high note + 0*low note;
for the second 1/10 of the high note
      combined note = 0.9*high note + 0.1*low note;
for the third 1/10 of the high note
      combined note = 0.8*high note + 0.2*low note;
      .
      .
      .
for the last 1/10 of the high note
      combined note = 0.1*high note + 0.1*low note;
for the rest of the samples
      combined note = 0*high note + 1* low note;
```

Figure 6. Pseudocode for the crossfading algorithm.

One must keep in mind that the pitched-down *high* note is significantly shorter in duration than the pitched-up *low* note; the crossfading algorithm utilizes the attack of the *high* note and gradually makes the transition to the decay of the *low* note, thus emulating the duration of the desired note as well.

Note that all calculations are done in a single-sample basis, while the audio buffer is also 'fed' one sample at a time. With this technique, the user can modify the pitch of each drum in real time without any audible artifacts or abnormalities in the produced note.

4.3.5 Damping

It is very hard to simulate a damped note from an existing recording of a 'normal' note. Seeing as our soundbank was increasing in a rapid manner, we decided to omit the choice of damped notes or mutes for the time being. However, this is an important feature and methods to include it are currently under investigation.

5. IMPLEMENTATION OF THE MODEL

Our final Timpani model consists of a pressure-sensitive digitizer tablet as the surface for all five timpani, using the stylus as a mallet and the tablet's scroll wheels as tuning cogs. The model's soundbank consists of 750 recorded samples, that feature 5 dynamic values, 5 hit points, 3 mallet types, and 2 notes for each of the five different kettledrums.

5.1 Sample acquisition

All samples were recorded at a conservatorium, using a five-piece set of Adams symphonic timpani. In order to control the exact dynamic and hit position for each sample, a simple mechanism was constructed (see Fig. 7); the mechanism consisted of an elastic length of metal suspended over the drum's head, on which the mallet was attached.

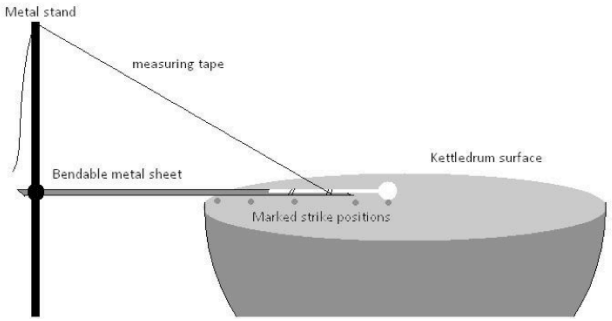


Figure 7. The mechanism created to control hit strength and position.

Each sample was given a unique ID number, in order to simplify the loading and management of samples. The ID was created with the following format: 1000*[type of mallet] - 0 for soft, 1 for hard and 2 for wooden mallets; 100*[number of drum] – 0 for the 20'' drum, 1 for the 23'' drum etc; 10*[dynamic value] – 0 for pianissimo, 2 for piano etc; and finally, 1*[hit position] – 0 for the center of the drum, 1 for staccato, and so on. For example, sample 2431 is a strike on the *staccato* position, with a *mezzoforte* dynamic value, on the *32-inch* drum with *hard* mallets.

5.2 Coding language & Audio processing API

The two main programming languages considered for the project implementation were Java and C++. Due to limited tablet support in Linux-based systems and relatively high latency in sample pre-mixing and event processing in Java, a Win-32 C++ implementation was chosen.

Our API of choice was the *Synthesis Toolkit* (STK) [15] in combination with *RtAudio* [16], due to its low-level functionality, versatility, and low latency performance. Moreover, STK and RtAudio are cross-platform; this allows for future porting of this project in order to make it available in Linux- and OSX-based systems.

5.3 Memory management

Currently, every sample is loaded in the memory during the program's startup; unfortunately, this way the application takes up to 1.100 megabytes of memory. However, it is a very easy task to modify the application so it only loads the samples needed for the mallet type chosen, thus reducing its memory footprint to 350 megabytes of RAM. Since the project intended to give the player full control over the timpani set as a timpanist would have, it was deemed important that the user could

switch between mallets without any delay or additional loading.

5.4 User Interface

Hit coordinates and strength are retrieved through the input provided by the tablet. For visualization purposes, a printed sheet depicting the position of each drum is placed on the tablet, as seen in Figure 8.

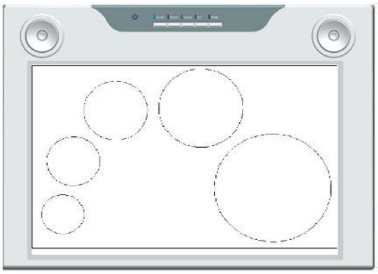


Figure 8. An example of the printed sheet used to visualize the position & size of each kettledrum.

A significant drawback of the control scheme is the fact that tablet drivers only allow for one stylus to be used. We bypassed this restriction by applying a simple solution; when the stylus hits the tablet, the first note is produced. For as long as the stylus stays pressed on the tablet, the x-axis direction in which it is moving is recorded; when the direction changes, a new note is produced. This way, the user can control the speed and dynamics of the timpani roll by 'drawing' smaller or wider circles on the tablet surface.

The system uses the default Windows libraries for tablet devices that ships bundled with all tablet drivers, thus eliminating the need for specific libraries. In the event that a tablet does not have scrolling wheels, the mouse wheel can be used to tune the each kettledrum in real time. Finally, the selection of mallet type is available to the user through the application's GUI.

6. RESULTS

6.1 Synthesis quality

The spectrograms of three real and synthesized notes can be seen in Figures 9-11. The synthesized notes were produced by the tuning algorithm presented in 4.3.4, using A2 and E2 as the low and high note respectively. As it can be seen, the overall structure and evolution of the harmonic series is faithful to the recorded notes; however, in cases such as the D2 note it can be seen that the higher harmonics are slightly enhanced.

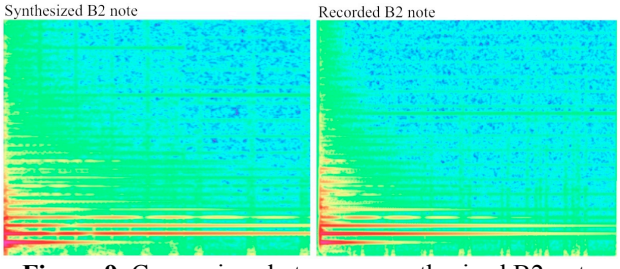


Figure 9. Comparison between a synthesized B2 note played on a 29'' kettledrum, and the real note.

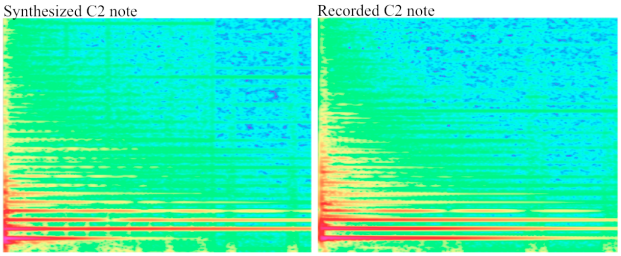


Figure 10. Comparison between a synthesized C2 note played on a 29'' kettledrum, and the real note.

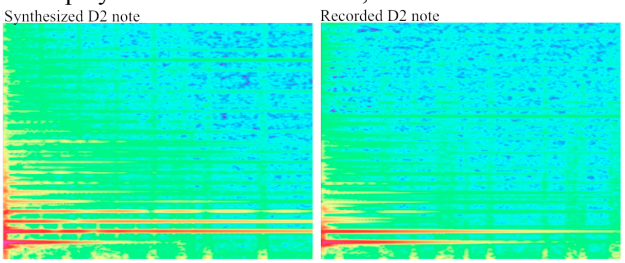


Figure 11. Comparison between a synthesized D2 note played on a 29'' kettledrum, and the real note.

An audiovisual demonstration of the model will soon be publically available on the Internet for evaluation purposes, on the address provided in [17].

6.2 Audio Latency

The real-time response of the model yields an audio latency of 25 milliseconds when tested with a bulk Realtek on-board audio card using the DirectSound audio driver. For an external audio card using an ASIO driver, audio latency is below 12 milliseconds.

7. CONCLUSIONS & FUTURE WORK

In this paper, we presented a working system that implements a real-time model of the Orchestral Timpani, controlled by a digitizer tablet. The user has control over almost every parameter that defines and shapes the sound of the instrument, and these parameters can be modified and adjusted in real time. Our results demonstrate that we successfully emulate the sound of the Orchestral Timpani, while achieving low latency.

On the other hand, there are certain drawbacks to our approach, the first of which is its rather high memory demands. A compromise would be to only keep the samples for one type of mallets loaded in the memory, imposing a brief loading time when the user chooses another type of mallets. However, a more efficient memory management scheme could be utilized in order to constantly maintain all samples loaded in the memory.

Another drawback is the fact that tablets only allow for one stylus to be used. This has its toll on the interaction scheme, as the instrument was designed to work with two mallets. Moreover, tablets do not provide tactile feedback of any kind, a characteristic which is inextricably linked with the process of playing percussion instruments; to this end, multi-touch input interfaces that provide tactile feedback to the user must be investigated in the future.

Finally, the model used in this project applies to many percussion instruments sensitive to different hit coordinates & hit strengths such as cymbals, bass drums, and gongs. If a new soundbank was recorded to feature such instruments, almost all algorithms and methods created for this project could easily be modified to include such instruments.

8. REFERENCES

[1] J.O. Smith: "Virtual Acoustic Musical Instruments: Review and update", *Journal of New Music Research* 33(3), pp. 283-304, 2004.

[2] S.V. Duyne, J.O. Smith: "The 2-D Waveguide Mesh", *IEEE Workshop on In Applications of Signal Processing to Audio and Acoustics*, pp. 177-180, 1993.

[3] L. Savioja, V. Välimäki: "Reduction of the dispersion error in the triangular digital waveguide mesh using frequency warping," *IEEE Signal Processing Letters*, vol. 6, no. 3, pp. 58–60, 1999.

[4] S. Bilbao: "A modular percussion synthesis environment", *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, 2009

[5] R. Jones, A. Schloss: "Controlling a Physical Model with a 2D Force Matrix", *Conference on New Interfaces for Musical Expression*, New York, 2007.

[6] L. Trautmann, S. Petrausch, R. Rabenstetin: "Physical Modeling of drums by transfer function methods", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 5, pp. 3385-3388, 2001.

[7] L. Trautmann and R. Rabenstein: "Digital Sound Synthesis by Physical Modeling Using the Functional Transformation Method". New York: Kluwer Academic, 2003.

[8] R. Marogna, F. Avanzini: "Physically-based Synthesis of Nonlinear circular membranes", *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, 2009

[9] F.Avanzini, R. Marogna: "A Modular Physically Based Approach to the Sound Synthesis of Membrane Percussion Instruments", *IEEE Transactions on Audio, Speech, and Language Processing*, 18(4), 2010

[10] R. E. Davis, "Mathematical modeling of the orchestral timpani", *Ph.D thesis*, Physics Department, Purdue University, 1988.

[11] P. Joly A. Chaigne, L. Rhaouti: "Time-domain modelling and numerical simulation of a kettledrum", *Journal of the Acoustic Society of America* 105(6), pp. 3545-3562, 1999.

[12] S. Emsen, C. Heal, R. Sunderland, R. Willis: "GDP 7 Electronic Timpani", University of Southampton - Faculty of Engineering and Applied Science, 2002.

[13] C. R. Nave: "Hyperphysics – The Timpani", http://hyperphysics.phy-astr.gsu.edu/hbase/music/timpani.html, retrieved on April 2010.

[14] Synesthesia Corp, "The Mandala Drum", http://www.synesthesiacorp.com/, retrieved on April 2010.

[15] P.R. Cook, G.P. Scavone: "The Synthesis Toolkit in C++", https://ccrma.stanford.edu/software/stk/, retrieved on April 2010.

[16] G.P. Scavone: "The RtAudio API", http://www.music.mcgill.ca/~gary/rtaudio/, retrieved on April 2010.

[17] http://www.youtube.com/user/kettleVirtualTimpani

TIMBRE REMAPPING THROUGH A REGRESSION-TREE TECHNIQUE

Dan Stowell and Mark D. Plumbley
Centre for Digital Music, Queen Mary University of London, UK
dan.stowell@elec.qmul.ac.uk

ABSTRACT

We consider the task of inferring associations between two differently-distributed and unlabelled sets of timbre data. This arises in applications such as concatenative synthesis/ audio mosaicing in which one audio recording is used to control sound synthesis through concatenating fragments of an unrelated source recording. Timbre is a multidimensional attribute with interactions between dimensions, so it is non-trivial to design a search process which makes best use of the timbral variety available in the source recording. We must be able to map from control signals whose timbre features have different distributions from the source material, yet labelling large collections of timbral sounds is often impractical, so we seek an unsupervised technique which can infer relationships between distributions. We present a regression tree technique which learns associations between two unlabelled multidimensional distributions, and apply the technique to a simple timbral concatenative synthesis system. We demonstrate numerically that the mapping makes better use of the source material than a nearest-neighbour search.

1. INTRODUCTION

This paper aims to improve musical expression by audio-based control of timbre. There are various applications in which the timbral analysis of a sound is used to control a system whose output is sound of a different type, such as concatenative synthesis [1][2][3], query-by-example [4] or adaptive digital audio effects [5]. In such cases there are two different timbral distributions to consider – that of the controlling sound, and that of the audio output – and we wish to be able to map from one to the other. Often the two distributions are quite different, since for example we may wish to map from vocal sounds on to timbres which cannot be produced vocally, so the issue of mapping is non-trivial.

In this paper we argue that mapping through standard techniques such as nearest-neighbour search may be insufficient, and we present a new nonparametric technique based on regression trees which accounts for the differences between timbral distributions. We then apply the technique in a simplified timbral concatenative synthesis system, and demonstrate numerically that the mapping makes better use of the source material than a nearest-neighbour search.

1.1 Timbre trajectories: absolute or relative?

In order to map the timbre trajectory of a sound (its evolution over time) onto another, we will require some timbre analysis of the signal. An issue that affects our choice of search strategy is whether the timbral analysis should best be treated as absolute context-independent data, or whether it should be treated as relative – for example, relative to the range of the sound source which produced it. Given a particular timbral "coordinate", should we treat it differently if we knew that it was produced by a clarinet or by a violin? Would such information imply a difference in the expressive purpose of the sound?

The common definition of timbre describes it as that attribute which enables a listener to differentiate sounds which are equal in pitch and loudness [6]. It therefore does not demand that timbre be an absolute or context-invariant attribute of a sound. Research into music timbre perception has taken a similar stance, basing experiments on comparisons among sets of sound examples [7, 8, 9, 10]. Such studies often explain results in part through acoustic features derived from the examples, which can imply a context-independent notion of timbre inherent in the signal. However Grey [11] finds evidence for context-dependence of timbre perception in musical patterns. Lakatos [12] offers some consideration of contextual effects by investigating sets of harmonic and percussive sounds both separately and combined. He presents evidence supporting the existence of two broadly context-independent timbre dimensions but also for some degree of contextual influence on timbre judgements.

Musical applications of timbre analysis often use acoustic features taken from the signal (e.g. [13][14, Chapter 16]), implicitly treating timbre as absolute. This will certainly be appropriate in situations where the timbre data contains strong semantic "anchors" – a clear example of this occurs in human speech, where vowels are largely characterised by the absolute positions of the main resonances (formants) on the frequency scale [15]. However, the evidence of context-dependence in musical timbre suggests this may not always be the case. Consider a system which synthesises sound based on timbral examples produced by voice (e.g. [14, Part III]): the human voice is naturally constrained to its own timbre range, yet we may well wish to induce the system to produce sounds outside this range. In fact we consider this to be a basic requirement, since such ability to extend our timbral range is one of the main ap-

peals of such technologies.

1.2 Timbre lookup strategies

The most basic form of timbral search is perhaps a nearest-neighbour (NN) search [16], often using Euclidean distance. Since timbre features in general have quite different ranges, their ranges may be standardised before search, or a scale-invariant metric such as Mahalanobis distance may be used [17]. For example, Schwarz [14, Chapter 16] uses the Euclidean distance normalised over the entire database of sounds. This normalisation accounts for differences between the ranges of the features, but not for differences between the timbral range of the different sound sources included in the database. Note that timbral distance search is but one criterion used in a concatenative synthesiser such as this, which uses a constraint-satisfaction framework to combine criteria related to duration, pitch and other considerations.

Large database search systems often do not store the raw timbral co-ordinates needed for NN search, but parametrically model the timbre of a recording (e.g. using Gaussian Mixture Models) and store the model parameters [13]. Timbral search can then be performed by finding the parameter-set which maximises the likelihood of query data.

Whether search is performed by instance-based methods such as NN or model-based methods such as Gaussian Mixture Model likelihood, the difference in timbral ranges of different sound sources is often neglected, perhaps reflecting an approach to timbre as absolute rather than relative. One approach to account for this could be to standardise the mean and variance of timbre features for each type of sound source, or for each recorded audio excerpt, which would accommodate the large-scale differences. However it would fail to account properly for multidimensional interactions in the data such as the movement of one region relative to the rest of the distribution.

Rather than pursuing the idea of a normalisation scheme as a precursor to search, in this paper we develop an integrated method which automatically learns to map from one data distribution to another, assuming similarities in the orientation of the datasets in timbre space but allowing for differences in the distributions at large and small scales. Tree methods are attractive in this context because recursive partitioning provides a generic approach to dividing multidimensional distributions into regions of interest at multiple scales. We next describe the method, before applying it in a concatenative synthesis experiment.

2. AUTO-ASSOCIATIVE REGRESSION TREES

A regression tree [18, Chapter 8] is a computationally efficient nonparametric way to analyse structure in a multivariate dataset, with a continuous-valued response variable to be predicted by a set of independent variables. The core concept is to recursively partition the dataset, at each step splitting it into two subsets using a threshold on one of the independent variables (i.e. a splitting hyperplane orthogonal to one axis). The choice of split at each step is made to minimise an "impurity" criterion for the value of the response variable in the subsets, often based on the mean squared error [18, Section 8.3]:

$$\text{impurity}(\alpha) = \sum_{i=1}^{n_\alpha} (y_i - \bar{y})^2 \tag{1}$$

where n_α is the number of data points in the subset α under consideration, and $\bar{y}$ the mean of the sampled values of the response variable y_i for the points in α.

The original formulation of regression trees was concerned with predicting a single univariate response variable. They were subsequently extended to multivariate responses, for example by [19] using a direct multivariate extension of (1):

$$\text{impurity}(\alpha) = \sum_{i=1}^{n_\alpha} \sum_{j=1}^{p} (y_{ij} - \bar{y}_j)^2 \tag{2}$$

with definitions as in (1) except that the y_i (and therefore also $\bar{y}$) are now p-dimensional vector values, with j indexing over the dimensions.

This extension yields a framework that can learn to infer relationships between one multivariate data distribution (the independent variables) and another (the response) – hence their potential application to the inference of mapping from one set of multivariate timbre data to another. One limitation of this is that the regression is still a supervised technique, meaning that the pairwise association between items in the training datasets would need to be provided. In applications such as ours, where we might have a large database of short audio fragments from various sources, it will often be impractical to annotate the data, so we seek an unsupervised method. We will develop an existing unsupervised application of regression trees for this task.

2.1 Auto-association and multivariate splits

Questier et al. [19] apply regression trees to the task of discovering structure in unsupervised multivariate data, by equating the response variables with the independent variables, to create an *auto-associative* multivariate regression tree (AAMRT). In other words they apply a standard regression tree with the multivariate-response extension, but there is no separation between the variables used to split the dataset and the variables whose impurity is to be minimised – the independent variables are made to "predict themselves". This is reminiscent of data-driven histogramming [20]; in the work of Questier et al. it is used for feature-selection by analysing which features are most commonly used for splitting.

There are in fact two types of multivariate extension to the standard regression tree.We have already described the *multivariate-response* extension; also the choice of splitting plane can be generalised so that it can take any orientation in the feature space rather than being aligned with one axis [21]. We refer to this as the *multivariate-splits* extension. Gama [22] shows that this extension can reduce bias in the resulting estimator. Further, it may make more

effective use of the available information if there is a limited number of datapoints: if there are N data points then there can be no more than around $\log_2 N$ splits used to reach a leaf in a balanced binary tree. This could well be fewer than the number of dimensions, meaning the information from some dimensions would be neglected. For the remapping task discussed in the next section, then, we will use a regression tree that is based on AAMRT but multivariate in both senses.

Note that the impurity measures (1) & (2) are equivalent to the sum of variances in the subsets, up to a multiplication factor which we can disregard for the purposes of minimisation. By the law of total variance (see e.g. [23, Appendix S]), minimising the total variance within the subsets is the same as maximising the variance of the centroids; therefore the impurity criterion selects the split which gives the largest difference of the centroids of the response variable in the resulting subsets. If only univariate splits are allowed then this can be optimised as given in [18, Chapter 8]. In the multivariate-splits variant, maximising the variance of the centroids is achieved simply by selecting the hyperplane perpendicular to the first principal component in the (centred) data. This multivariate-splits variant of AAMRT allows for efficient implementation since the leading principal component in a dataset can be calculated efficiently e.g. by expectation-maximisation [24].

3. CROSS-ASSOCIATION

We wish to generalise the AAMRT method to apply it to two datasets defined on the same space. A simple approach would be to combine the two datasets into one and then apply AAMRT, but this would not allow the algorithm to adapt separately to the two datasets, to account for differences in location.

Instead, we modify the algorithm so that at each step of the recursion the data coming from the two distributions are separately centred. One single principal component is then calculated from their union. The recursion therefore generates two "similar but different" trees, implementing the notion that the two datasets have similarities in structure (the orientations of the splitting planes are the same) but may have differences in location at various scales (the centroids of large or small subsets of the data are allowed to differ). This is illustrated schematically in Figure 1.

If the datasets are unequal sizes then the larger set will tend to dominate over the smaller in calculating the principal component. To eliminate this issue we weight the calculation so as to give equal emphasis to each of the datasets, equivalent to finding the principal component of the union J of weighted datasets:

$$J = \bigcup (N_Y(X - C_X),\ N_X(Y - C_Y)) \qquad (3)$$

where X and Y represent the data (sub)sets, C_X and C_Y their centroids, and N_X and N_Y the number of points they contain.

The resulting cross-associative multivariate regression tree (XAMRT) algorithm is summarised in Figure 2. Note that we do not prune the tree [18, Chapter 3], since for the

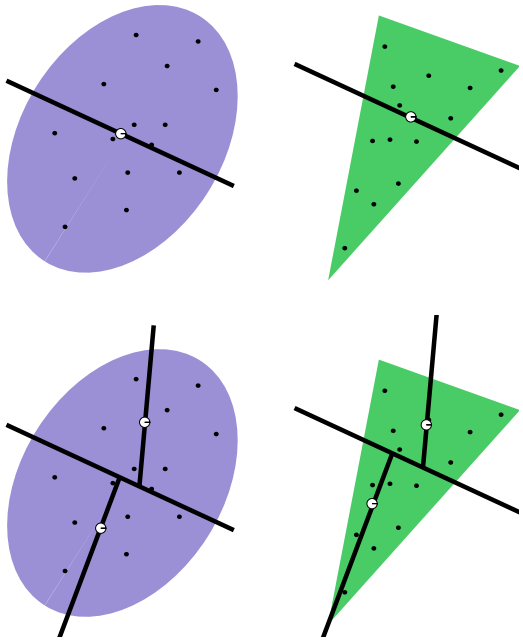


Figure 1. Schematic representation of the first two steps in the recursion. In the first step (top), the centroids of each dataset are calculated separately, and then a splitting plane with a common orientation is chosen. The second step (bottom) is the same but performed separately on each of the partitions produced in the first step.

timbral application presented here, all of the variation in the training set is useful for resynthesis.

To perform a remapping using a XAMRT data structure, one takes a data point and descends the tree, at each split centring it by subtracting C_X or C_Y as appropriate and then deciding which side of the splitting plane it falls. When the leaf node is reached, it contains two sets of training data points (a subset each of X and Y). To choose a corresponding coordinate relating to the opposite distribution, one could for example use a random datum selected from the opposite subset, or the centroid of that subset, depending on the application. (If the sizes of the datasets are similar then the leaf will often contain just one datum from each of the two distributions.)

4. TIMBRE REMAPPING EXPERIMENT

Our algorithm can be applied to timbre remapping tasks, i.e. ones in which the timbral trajectory of a sound source is used to control that of some other system. Concatenative synthesis is an example of such a task, in the case where an input sound is used as the controller for the concatenative synthesiser (also referred to as audio mosaicing). Concatenative synthesis is not the only example of timbral application of our algorithm – we are investigating application of the technique in general synthesiser control – but it presents a known system in which timbres from heterogeneous sources are used to control sound generation.

Numerical evaluation of timbre remapping quality is difficult since the perceptual quality and musical merit of the audio result have no obvious objective metric. How-

XAMRT(X, Y)

$C_X \leftarrow$ centroid of X
$C_Y \leftarrow$ centroid of Y
$J \leftarrow$ result of equation (3)
$p \leftarrow$ principal component of J
$X_l \leftarrow X \cap ((X - C_X) \cdot p > 0)$
$X_r \leftarrow X \cap ((X - C_X) \cdot p \leq 0)$
$Y_l \leftarrow Y \cap ((Y - C_Y) \cdot p > 0)$
$Y_r \leftarrow Y \cap ((Y - C_Y) \cdot p \leq 0)$
if X_l is singular or Y_l is singular
  then $L = [X_l, Y_l]$
  else L = XAMRT(X_l, Y_l)
if X_r is singular or Y_r is singular
  then $R = [X_r, Y_r]$
  else R = XAMRT(X_r, Y_r)
return $[L, R]$

Figure 2. The cross-associative algorithm. X and Y are the two sets of vectors between which associations will be inferred.

Description	Duration (sec)	No. of grains
Amen breakbeat	7	69
Beatboxing	93	882
Fireworks	16	163
Kitchen sounds	49	355
Thunder	8	65

Table 1. Audio excerpts used. "No. of grains" is the number of 100ms grains segmented and analysed from the audio (excluding silent frames) – see text for details.

ever, concatenative synthesis offers the opportunity for numerical evaluation by studying the statistics of usage of the different grains, as will be described in Section 4.4.

We require an experiment which will probe the timbral matching performance of our algorithm. Concatenative synthesisers typically operate not only on timbre, but use pitch and duration as well as temporal continuity constraints in their search strategy, and then modify the selected grains further to improve the match [25]. While recognising the importance of these aspects in a full concatenative synthesis system, we designed an experiment in which the role of pitch, duration and temporal continuity were minimised, by excluding such factors from grain construction/analysis/resynthesis, and also by selecting audio excerpts whose variation is primarily timbral.

We first describe the audio excerpts we used and how timbre was analysed, before describing the concatenative synthesiser and our performance metric.

4.1 Audio data

In order to focus on the timbral aspect, we selected a set of audio excerpts in which the interesting variation is primarily timbral and pitch is less relevant. The five excerpts – two musical (percussive) and three non-musical – are listed in Table 1 and are also available online. [1] The excerpts are 44.1 kHz mono recordings.

The excerpts are quite heterogeneous, not only in sound source but also in duration (up to an order of magnitude). They each contain various amounts/types of audio event, which are not annotated. This wide variety of excerpts was selected to give a clear impression of the success of the remapping techniques at drawing timbral analogies.

4.2 Timbre features

We chose a set of 10 common acoustic timbre features: spectral power, spectral power ratio in 5 log-spaced subbands (50–400, 400–800, 800–1600, 1600–3200, and 3200–6400 Hz), spectral centroid, spectral 95- and 25-percentiles and zero-crossing rate (for definitions see [26]).

Analysis was performed on audio "grains": units of fixed 100ms duration taken from the audio excerpt every 100ms (i.e. with no overlap). Each grain was analysed by segmenting into frames of 1024 samples (at 44.1 kHz sampling rate) with 50% overlap, then measuring the feature values for each frame and recording the mean value of each feature for the grain. Grains with a very low spectral power (< 0.002) were treated as silences and discarded. The timbre features of the remaining grains were normalised to zero mean and unit variance within each excerpt. Analysis was performed in SuperCollider 3.3.1 [27].

4.3 Timbral concatenative synthesiser

We designed a simple concatenative synthesiser using only timbral matching, either by a standard nearest-neighbour (NN) search or by our algorithm. Given two excerpts – one which is the source of grains to be played back, and one which is the control excerpt determining the order of playback – and the timbral metadata for the grains in the two excerpts, the synthesis procedure works as follows:

For each grain in the control excerpt, if the grain is silent (power < 0.002) then we replace it with silence. Otherwise we replace it with a grain selected from the other excerpt by performing a lookup of the timbre features – either a NN search or the XAMRT tree regression. For numerical evaluation, the choice of grain is recorded. For audio resynthesis, the new set of grains is output with a 50ms linear crossfade between grains.

The NN search uses the standard Euclidean distance, facilitated using a k-d tree data structure [28]. Note that the timbre features are normalised for each excerpt, meaning the NN search is in a normalised space rather than the space of the raw feature values.

In both the NN and XAMRT lookup there is an issue of tie-breaking. More than one source grain could be retrieved – at the minimum distance from the query (for NN) or in the leaf node retrieved from the query (for XAMRT) – yet we must choose only one. This is not highly likely for NN search (depending on the numerical precision of the implementation) but will occur in XAMRT when mapping from a small to a large dataset, since the tree can grow only to the size allowed by the smaller dataset. Additional

[1] `http://archive.org/details/xamrtconcat2010`

criteria (e.g. continuity) could be used to break the tie, but for this experiment we keep the design simple and avoid confounding factors by always choosing the grain from the earliest part of the recording in such a case.

4.4 Evaluation method

The ultimate evaluation of musical synthesis techniques is through listening tests; however we defer this to later work, when we plan to incorporate the technique into more complete synthesis systems. For development and comparison purposes it is particularly helpful to have objective measures of success. It is natural to expect that a good concatenative synthesiser will make wide use of the "alphabet" of available sound grains, so as to generate a rich as possible output from the limited alphabet. Here we develop this notion into an information-theoretic evaluation measure.

Communication through finite discrete alphabets has been well studied in information theory [29]. A key information-theoretic quantity is the (Shannon) *entropy*, defined for a discrete random variable X taking values from an alphabet $\mathcal{A}$ as

$$H(X) = -\sum_{i=1}^{|\mathcal{A}|} p_i \log p_i \tag{4}$$

where p_i is the probability that $X = \mathcal{A}_i$ and $|\mathcal{A}|$ is the number of elements in $\mathcal{A}$. The entropy $H(X)$ is a measure of the information content of X, and has the range

$$0 \leq H(X) \leq \log |\mathcal{A}| \tag{5}$$

with the maximum achieved iff X is uniformly distributed.

If the alphabet size is known then we can define a normalised version of the entropy called the *efficiency*

$$\text{Efficiency}(X) = \frac{H(X)}{\log |\mathcal{A}|} \tag{6}$$

which indicates the information content relative to some optimised alphabet giving a uniform distribution. This can be used for example when X is a quantisation of a continuous variable, indicating the appropriateness of the quantisation scheme to the data distribution.

We can apply such an analysis to our concatenative synthesis, since it fits straightforwardly into this framework: timbral expression is measured using a set of continuous acoustic features, and then "quantised" by selecting one grain from an alphabet to be output. It does not deductively follow that a scheme which produces a higher entropy produces the most pleasing audio results. However, a scheme which produces a low entropy will tend to be one which has an uneven probability distribution over the grains, and therefore is likely to sound relatively impoverished – for example, some grains will tend to be repeated more often than in a high-entropy scheme. Therefore the efficiency measure is useful in combination with the resynthesised audio results for evaluating the grain selection scheme.

Query type	Efficiency (%)
Nearest neighbour	70.8 ± 4.4
XAMRT	84.5 ± 4.8

Table 2. Experimental values for the information-theoretic efficiency of the lookup methods. Means and 95% confidence intervals are given. The improvement is significant at the $p < 0.000001$ level (paired *t*-test, two-tailed, 19 degrees of freedom, $t = 12.47$).

4.5 Results

We applied the concatenative synthesis of Section 4.3 to each of the 20 pairwise combinations of the 5 audio excerpts (excluding self-to-self combinations, which are always 100% efficient) using each of the two lookup methods (NN and XAMRT). We then measured the information-theoretic efficiency (6) of each run. Table 2 summarises the efficiencies for each lookup method. Our method is seen to be significantly better than the NN search, improving efficiency by over 13 percentage points.

Audio examples of the output are available online. [1] Note that the reconstructed audio examples sound rather unnatural because the experiment is not conducted in a full concatenative synthesis framework. In particular we use a uniform grain duration of 100ms and impose no temporal constraints, whereas a full concatenative synthesis system typically segments sounds using detected onsets and includes temporal constraints for continuity, and therefore is able to synthesise much more natural attack/sustain dynamics [25].

Such factors mean our audio outputs are tricky to judge by listening, and it is not quite clear how far the advantage in efficient use of grains translates into an improved perceptual richness of the output – i.e. into improvements in the timbral analogies made. Nevertheless, our method shows promise as the timbral component of a multi-attribute search which could potentially be used in concatenative synthesis, as well as other applications requiring timbral search from audio examples (e.g. query-by-example [4]).

5. CONCLUSIONS AND FURTHER WORK

We have developed a nonparametric technique able to learn associations from one unlabelled data distribution to another defined on the same space, assuming similarity in structure of the data distributions but accounting for differences in location and shape. This provides a robust and efficient way to map timbre trajectories from one sound source onto timbre trajectories to be performed with a different sound source, making good use of the timbral variation available in the latter. In experiments with a simplified concatenative synthesiser, we have demonstrated that it makes significantly better use of the source material than a nearest-neighbour search.

Future work would integrate this approach into a full concatenative synthesis framework, and supplement the objective tests with listening tests. We also intend to apply the technique to other types of synthesis to control them by audio input.

6. REFERENCES

[1] D. Schwarz, "Current research in concatenative sound synthesis," in *Proceedings of the International Computer Music Conference (ICMC)*, pp. 9–12, 2005.

[2] T. Jehan, "Event-synchronous music analysis/synthesis," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx-04)*, (Naples, Italy), pp. 361–366, 2004.

[3] B. Sturm, "Adaptive concatenative sound synthesis and its application to micromontage composition," *Computer Music Journal*, vol. 30, no. 4, pp. 46–66, 2006.

[4] M. LeSaffre, D. Moelants, M. Leman, B. De Baets, H. De Meyer, G. Martens, and J.-P. Martens, "User behavior in the spontaneous reproduction of musical pieces by vocal query," in *Proceedings of the 5th Triennial ESCOM Conference*, (Hannover, Germany), pp. 208–211, 2003.

[5] V. Verfaille, U. Zölzer, and D. Arfib, "Adaptive digital audio effects (A-DAFx): a new class of sound transformations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1817–1831, 2006.

[6] ANSI, *Acoustical Terminology*. No. S1.1-1960, New York: American National Standards Institute, 1960.

[7] J. M. Grey and J. W. Gordon, "Perceptual effects of spectral modifications on musical timbres," *Journal of the Acoustical Society of America*, vol. 63, no. 5, pp. 1493–1500, 1978.

[8] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: common dimensions, specificities, and latent subject classes," *Psychological Research*, vol. 58, no. 3, pp. 177–192, 1995.

[9] A. Caclin, S. McAdams, B. K. Smith, and S. Winsberg, "Acoustic correlates of timbre space dimensions: a confirmatory study using synthetic tones," *Journal of the Acoustical Society of America*, vol. 118, no. 1, pp. 471–482, 2005.

[10] J. A. Burgoyne and S. McAdams, "A meta-analysis of timbre perception using nonlinear extensions to CLASCAL," in *Sense of Sounds* (R. Kronland-Martinet, S. Ystad, and K. Jensen, eds.), vol. 4969/2009 of *Lecture Notes in Computer Science*, ch. 12, pp. 181–202, Berlin: Springer, 2009.

[11] J. M. Grey, "Timbre discrimination in musical patterns," *Journal of the Acoustical Society of America*, vol. 64, no. 2, pp. 467–472, 1978.

[12] S. Lakatos, "A common perceptual space for harmonic and percussive timbres," *Perception & Psychophysics*, vol. 62, no. 7, pp. 1426–1439, 2000.

[13] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: how high's the sky?," *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, 2004.

[14] D. Schwarz, *Data-Driven Concatenative Sound Synthesis*. PhD thesis, IRCAM, Paris, France, Jan 2004.

[15] D. Deterding, "The formants of monophthong vowels in Standard Southern British English pronunciation," *Journal of the International Phonetic Association*, vol. 27, no. 1, pp. 47–55, 1997.

[16] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Computing Surveys*, vol. 33, pp. 273–321, 2001.

[17] J. Wouters and M. W. Macon, "A perceptual evaluation of distance measures for concatenative speech synthesis," in *Proceedings of ICSLP'98*, vol. 6, (Sydney), pp. 2747–2750, Nov 1998.

[18] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth Statistics/Probability Series, Wadsworth Inc, 1984.

[19] F. Questier, R. Put, D. Coomans, B. Walczak, and Y. V. Heyden, "The use of CART and multivariate regression trees for supervised and unsupervised feature selection," *Chemometrics and Intelligent Laboratory Systems*, vol. 76, no. 1, pp. 45–54, 2005.

[20] G. Lugosi and A. Nobel, "Consistency of data-driven histogram methods for density estimation and classification," *Annals of Statistics*, vol. 24, no. 2, pp. 687–706, 1996.

[21] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees," *Machine Learning*, vol. 19, no. 1, pp. 45–77, 1995.

[22] J. Gama, "Functional trees," *Machine Learning*, vol. 55, no. 3, pp. 219–250, 2004.

[23] S. R. Searle, G. Casella, and C. McCulloch, *Variance Components*. Wiley-Interscience, online ed., 2006.

[24] S. T. Roweis, "EM algorithms for PCA and SPCA," in *Advances in Neural Information Processing Systems (NIPS)*, (Denver, Colorado), pp. 626–632, 1998.

[25] E. Maestre, R. Ramírez, S. Kersten, and X. Serra, "Expressive concatenative synthesis by reusing samples from real performance recordings," *Computer Music Journal*, vol. 33, no. 4, pp. 23–42, 2009.

[26] G. Peeters, "A large set of audio features for sound description," tech. rep., IRCAM, 2004.

[27] J. McCartney, "Rethinking the computer music language: SuperCollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[28] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[29] C. Arndt, *Information Measures*. Springer, 2001.

MELODIC MEMORY AND ITS DEPENDENCE ON FAMILIARITY AND DIFFICULTY

Mariana E. Benassi-Werke
Psychobiology Dept.
UNIFESP, Brazil
marianawerke@yahoo.com.br

Marcelo Queiroz
Comp. Science Dept.
USP, Brazil
mqz@ime.usp.br

Nayana G. Germano
Psychobiology Dept.
UNIFESP, Brazil
nayanager@hotmail.com

Maria Gabriela M. Oliveira
Psychobiology Dept.
UNIFESP, Brazil
mgabi@psicobio.epm.br

ABSTRACT

This paper addresses one aspect of human music cognition, which is the recollection of melodic sequences stored in short-term memory, and the manipulation of such items in working memory, by measuring spans of successfully recalled melodic sequences. In order to avoid long-term memory collaboration in this task, short-term memory measurements are made using randomly-generated melodic sequences, which in turn may sound difficult and unfamiliar to many experimental subjects. We investigate the dependence of melodic span measures on such aspects as familiarity and difficulty, both in direct-order recalling (as it relates to short-term memory capacity) and in inverse-order recalling (as it relates to working memory capacity). We also discuss the relation of these measurements to cognitive models of short-term and working memory for verbal and melodic material.

1. INTRODUCTION

Understanding human music cognition is a colossal task, which nevertheless must be undertaken. Besides its scientific interest per se, better understanding the way we humans process musical information should allow further developments in computational psychoacoustics, particularly in cognitive models for automatic feature extraction, with implications for both automatic musical analysis and computer-based sound synthesis.

This study is a small contribution to the understanding of one very restricted musical cognitive task, namely our ability to reproduce melodic fragments we never heard before. This ability involves a part of our cognition usually referred to as short-term memory, which has been extensively studied in the field of experimental psychology [1]. More recently, Baddeley and Hitch [2] proposed a refined model called *working memory*, that subsumed the notion of short-term memory, and eventually became the de facto standard model referring to short-term memory.

We wish to investigate the response level of our working memory to simple tasks such as reproducing a melodic fragment in direct order or in inverse order (also called reverse or retrograde, not to be confused with melodic inversion). Such an experiment is a straightforward transposition of classical experiments with working memory using sequences of digits or words, which in our case is aimed at identifying common or disparate elements in the processing of verbal and melodic information. As in the verbal case, random sequences should be used in order to avoid the contribution of long-term memory, which we routinely use in the memorization of whole musical pieces, for instance.

Random melodies can be quite hard to recall due to many concurrent factors, which might be empirically hypothesized, such as the number of distinct tones in a sequence, or the interval relations between adjacent notes. Similar factors might also affect the memorization of numbers or words, although interval relations may have no meaning in most non-musical contexts. Other factors, such as word-length and phonological similarity, are well-known to affect verbal memorization [3].

The difference in number and internal organization of distinct tones is also a characteristic feature of musical scales, such as the pentatonic (5-tone), diatonic (7-tone) and chromatic (12-tone) scales. Particularly in western musical education, diatonic and chromatic scales are everywhere present, from church modes through classical tonal music to 20th-century atonality. Yet the frequency with which diatonic scales have been employed in western folk, popular and classical music overshadows those relatively few pristine examples of entirely chromatic compositions. It is relatively safe to say that the average person growing up in western civilization is biased towards being more familiar with diatonic rather than chromatic musical examples.

Different interval relations between adjacent notes might also affect differently our perception, memorization and ability to reproduce melodic sequences. To name a few cases where such difference is mentioned, Fux's *Gradus ad Parnassum* of 1725 advises composers not to use large melodic leaps such as sixths or sevenths because they are hard to sing, and Nicola Vaccaj's *Metodo Pratico di Canto* of 1832 is arranged progressively according to melodic leaps. This suggests that smaller intervals (seconds and thirds) are easier (to sing) than larger intervals, and raises the question of whether they might also be easier to memorize.

Our main goal is to investigate the effects of familiarity and difficulty of melodies on our cognitive ability to reproduce and to retrograde such melodies at first hearing. It should be noted that we do not attempt to define the general

notions of familiarity and difficulty in music, but instead we identified two particular aspects that seem to capture a fragment of these general notions. By adopting diatonic and chromatic scales as representatives of more or less familiar melodic contexts, we are constraining our experiment in well-defined ways, enabling us to question whether (this aspect of) familiarity influences melodic span measures. Also, by comparing the memorization of melodies made up of only small intervals to general melodies without interval constraints we may have a glimpse at the effect of melodic difficulty on our working memory.

Although the answer to these questions may appear self-evident for a practising musician, we intend to give objective, experimental answers to these questions. These answers, it should also be noted, are assumed to depend on history and culture, and ours are no exception, since we work within the biased boundaries of our experimental population. Our efforts are not directed to uncovering universal or innate facts about human cognition, and we make no claim to universal validity. Any such claim would have to be verified by crosscultural or transhistorical experimentation.

Another goal of this text is to discuss the differences and similarities between verbal and melodic memorization, and their possible implications for the structure of the working memory model. By comparing performance measures, in both forward and backward span tests, for sequences of digits and tones, it is possible to better understand the underlying mechanisms that comprise working memory. Specifically, we add a few experimental facts to the discussion of whether there might be a separate short-term memory component for dealing with tonal information [4, 5, 6, 7].

This paper is organized as follows. We introduce the cognitive model of working memory in section 2, and describe the methodology for constructing and applying the experiment on human subjects in section 3. We discuss the computational analysis of experimental data, the statistical analysis for hypothesis testing, and the experimental results in cognitive terms in section 4, and final remarks and pointers to future research are given in section 5.

2. THEORETICAL FRAMEWORK

The working memory model proposed by Baddeley and Hitch [2] in 1974 consists of three interconnected components (see figure 1), namely the *central executive*, the *phonological loop* and the *visuospatial sketchpad*. The system formed by these interconnected components is supposed to account for short-term storage and real-time processing of incoming information, and is vital for higher cognitive functions such as reasoning, planning and communication. A fourth component named *episodic buffer* was later added by Baddeley [3], but its discussion lies outside the scope of this paper.

According to this model, the phonological (also called articulatory) loop is responsible for short-term storage of auditory information and is capable of maintaining items in memory, for instance by using a subvocal rehearsal process. The visuospatial sketchpad (or scratchpad) allows for

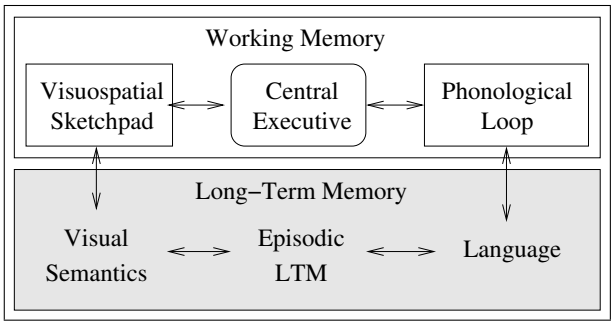


Figure 1. Baddeley and Hitch's Working Memory model.

temporary storage and manipulation of visual and spatial information. The central executive is the attentional focus of the system and is responsible for controlling and coordinating the other subsystems, allowing for the recollection of recent experience and the symbolic manipulation of recollected items [3]. Both visuospatial and phonological subsystems are supposed to interact with long-term memory components, such as language and visual semantics, that may aggregate meaning to items in working memory.

A verifiable characteristic of these subsystems is the fact that they have limited storage capacity, which can be measured by *digit span tests* [8]. These tests consist of presenting random digit sequences of increasing length and asking for immediate reproduction. The forward digit span of an individual is defined as the maximum length of a sequence he or she is able to correctly reproduce; usually two sequences are tried for each length, to account for slips of attention or other disturbing factors unrelated to memory capacity. The backward digit span reflects an individual's ability to correctly reproduce a sequence of digits in inverse order, and measures the working memory capacity of simple symbolic manipulation of recently-presented items.

One interesting aspect of span measures is the fact that they are highly dependent on several aspects of the nature of information being presented. For instance, digit span measures differ significantly across languages, probably due to differences in word size, phonetic similarity and semantic context [9, 10]. The effect of these differences can be examined by measuring memory spans for sequences of words of controlled size and phonetic content, or for sequences in different languages using bilingual subjects. For these subjects, the measured span of recollection of sequences of numbers or words in their first language is higher than spans in their second language [10, 11, 9]. According to Thorn & Gathercole [9], the maintenance in the phonological loop of familiar sound patterns of a well-known language benefits from lexical and sublexical knowledge to complement mental representations, and is thus more effective for the first language than for the second language in bilingual subjects. These results suggest that the storage of items in the phonological loop is influenced by semantic and phonological long-term memory, as proposed by Ardilla [10].

Although phonetic and semantic aspects have been extensively studied within the working memory model for verbal items, purely tonal information have received con-

siderably less attention in the literature [4, 5, 6, 7]. These works are concerned with *recognition tests*, where an isolated tone (stimulus) is presented and after a few seconds is compared to another tone (target). The insertion of irrelevant and unrelated tones between stimulus and target is known to degrade performance in tone recognition tests, whereas in digit recognition tests with irrelevant digits inserted between stimulus and target the decay in performance is barely noticeable [5]. This suggests that mechanisms of melodic and verbal storage might be independent.

One way of tackling this difficult question is by studying differences in memory span performance for melodic sequences both in direct and inverse order. One measure of performance degradation of backward spans with respect to direct span measures for the same information type is the *span index*, defined as the relative difference between forward and backward span measures. Different information processed by the same underlying mechanism would probably suffer from comparable degradation when passing from forward spans to backward spans, whereas significant differences in span index suggest that the underlying mechanisms might be different.

3. EXPERIMENTAL METHODOLOGY

In this section we present the methodology used in our experiments. The level of details offered should enable the realization of similar experiments with different populations and the comparison of both quantitative and qualitative results. We discuss the methodology in three stages. In section 3.1 we discuss the generation of the melodic sequences with varying levels of difficulty and familiarity (as discussed in section 1). In section 3.2 we discuss the prerequisites for individuals participating in the experiment, and also the first steps in selecting a reasonable population. The final application of the span tests is discussed in section 3.3.

3.1 Sequence Generation

The generation of data used in the melodic span tests is a crucial step in setting up the experiment, because the several sequences should reflect the relevant aspects of the questions we would like to answer. As discussed in section 1, we want to compare the difference in span performance in a more familiar and in a less familiar melodic context, as well as in a constrained, less difficult interval context and in an unconstrained, more difficult interval context. The defining attributes for these musical contexts in this particular experiment is as follows:

Familiarity: sequences are generated either within a single diatonic scale (e.g., C major) or within a chromatic scale.

Difficulty: subsequent tones in a sequence are generated either with constrained intervals (up to a major third upwards or downwards) or without any interval constraints.

These categories might be easily extended to consider other scales (e.g., pentatonic or quarter-tone scales) and other levels of difficulty (e.g., leaps up to a fifth or up to an octave), but the duration of the tests increase correspondently, and can easily become unbearable for the experimental subject. The average duration of the current experiment for each subject was about 90 minutes.

For each of the four combined contexts (more/less familiar and more/less difficult) a list of sequences of ascending length is generated, starting with 2 notes and going up to 10 notes, and always in pairs (2 sequences with N notes, for N=2,. . .,10). Since these tests require the subject to sing a melodic sequence, care should be taken with respect to the range of allowable tones. Each individual voice has its own tessitura, but in order to achieve uniformity of data and results some sort of compromise must be reached. We adopted a common range for female voices of [C4. . .C5], that correspond roughly to the intersection of soprano and alto registers (considering non-professional singers), and correspondingly the range of [C3. . .C4] for male voices. This corresponds to using up to 8 distinct tones in diatonic sequences and up to 13 distinct tones in chromatic sequences.

In order to be able to compare the effects of these contexts to what happens in similarly constrained verbal contexts, each melodic sequence was used to create a corresponding numerical sequence, by adopting the translations C4=1, D4=2, . . ., C5=8 for diatonic sequences and C4=1, C#4=2, . . ., C5=13 for chromatic sequences (and analogously for male voices). This way, we may also verify whether such restrictions on the number of symbols and internal structure of the sequences have some impact in span performance measures of numerical sequences.

Three additional constraints that appear in digit span tests were added in the sequence generation in order to avoid redundant sequences, which might be easier to recall due to the effect of *chunking* [3]:

- tones in a sequence can only reappear if strictly necessary (i.e., if sequence length $>$ # of distinct tones used), and in such case there should be at least four distinct intermediate tones between repeated tones.
- sequences with large monotonic subsequences (e.g., 5 or more successive upward or downward steps) or with few direction changes (e.g., less than 2 breakpoints in a sequence with 7 or more tones) should be discarded.
- sequences with a large common subsequence (N$\geq$3) with respect to the previous sequence in the same set (or a large repeating subsequence within it) should also be discarded.

Sequences to be used in inverse order were independently generated, instead of reusing direct order sequences, to avoid long-term memory collaboration. A total of 144 melodic sequences were thus generated, and the same amount of numerical sequences were obtained by direct translation.

Subsequently, all sequences were converted to audio, to guarantee that every individual is exposed to the same stimuli. Tones were synthesized as suggested in [7], by

using plain sine waves, with 0.1 sec fade-in and fade-out ramps and a total duration of 0.5 sec per tone, followed by 0.5 sec of silence. Numbers were recorded using both female and male voices and sequenced in order to keep the same duration of 1 sec between the starts of consecutive numbers.

3.2 Population Requirements

A first requirement for any individual participating in this experiment was already stated in the previous section. Since responses are collected via singing, the individuals have to be able to sing; more precisely, we need to be sure that each individual participating in the experiment has the ability to hear a tone and reproduce it correctly, within a reasonably defined tolerance.

For our experiment we considered a population of volunteers that consisted of amateur choir singers and music undergraduate students. This may be viewed as a heterogeneous population, since they show significant differences in musical background, singing skills and even musical memory skills (since those without sight-reading skills usually rely only on their memory for acquiring repertoire). With all their diversity, they generally satisfy the two most important aspects in defining the population for this experiment, which are (1) the common exposure to western popular and classical music and (2) the ability to sing in tune.

We defined a tuning test to be applied before the actual span tests, which consisted of hearing tones and reproducing each one immediately after hearing it (no sequence memorization required). We used a 12-tone row (taken from Schoenberg's Variations Op. 31) for this test, and only individuals who reproduced the 12 tones correctly would be considered for the final experiment.

The tolerance used to decide whether a tone has been correctly reproduced is also a critical point of the experiment. It should be noted that the experiment tries to grasp something that lies inside the subject's mind (i.e., in his memory), but the empirical data is modulated by his/her vocal skills. In an attempt to overcome this difficulty, we accept as correct any tone within a quarter-tone distance from the target, even if just in passing (in the case of an unstable vocal emission). More details are given in section 4.1.

3.3 Span Tests

An experimental session consists of a short explanation about the nature of the experiment and the format of the tests, after which the volunteer reads and signs a written informed consent to become part of the experimental research. This is then followed by the tuning test, and after that the actual span tests. The application of the 144 melodic span tests (plus 144 numerical span tests [1]) as defined earlier is divided into categories of similar data in ascending length order, such as "random diatonic sequences with restricted intervals in direct order", and so on. The ordering of these categories needs to be balanced, by using several distinct permutations of the categories, in order to cancel out the effects of fatigue and progressive familiarization of experimental subjects with the tests.

As in the case of classic digit span tests, each category of sequences of ascending length has 2 distinct sequences for each length value, and the span of a subject for that particular category is defined as the largest length N for which the subject correctly reproduces at least one of the sequences for all lengths up to N. This flexibility is supposed to account for distraction, singing mistakes, and other disturbing factors not necessarily related to an individual's working memory system.

Forward span measures correspond to span values for sequences that were supposed to be reproduced in direct order (i.e., as heard). Analogously, *backward span measures* correspond to span values for sequences that were supposed to be mentally reversed by the subject before being sung back.

The presentation of stimuli is always made using headphones to minimize external interference, and all responses are recorded using a microphone. In our tests, stimuli were organized for individual presentation in a computer with an external M-Audio MobilePre USB soundboard, and all recordings were made with a Samson C15 Studio condenser microphone using 16 bit samples and 44.1 kHz sampling rate.

4. EXPERIMENTAL RESULTS

The experiment described in the previous section was conducted with 13 volunteers, 8 amateur choir singers and 5 music undergraduate students. Of these, 10 volunteers passed the tuning test and were used in the analysis. The other 3 volunteers were amateur choir singers who had borderline tuning results (exactly 1 tone off by a semitone) and were discarded. Such borderline results might be attributed to distraction or other factors unrelated to perceptual or singing skills, and may be futurely included in analysis as a separate population.

The recordings were automatically analyzed and semi-automatically graded, giving a span measure for each individual and for each category of sequences (as discussed in the previous section). These measures were then submitted to statistical Analysis of Variance (ANOVA), out of which our original hypotheses were put to test. These steps and some cognitive remarks are given in the subsequent sections.

4.1 Analyzing the Recordings

All recordings were analyzed using a monophonic transcription audio system specially tailored for this experiment. The application context departs from traditional automatic transcription problems in several aspects, such as irrelevance of precise rhythmic information, a priori knowledge of timbral structure (voice), and silence as a separator of relevant events, to name a few. This characterizes a relatively simpler transcription subproblem, which is solved

[1] We use the term *numerical span* instead of the usual *digit span* because in our sequences items may be composed of two digits, and this is likely to affect span measures causing them to differ from well-known digit span values.

by a four-step method described below.

The first step of the analysis was based on [12]. Recorded signals were divided in frames of 1024 samples and a peak-detection strategy was applied for each frame, creating a set of candidate spectral peaks. The accuracy of peak estimates was improved using signal derivatives [13], and F0 estimates for each frame were obtained by maximizing the cumulative harmonic energy of each candidate peak [12].

The second step of the analysis consisted in filtering out spurious results of the first step by median filtering F0 values and subsequently marking nearly-silent frames as event separators. This step produced a nearly stable F0 profile for each isolated utterance.

The third step consisted in transcribing these F0 profiles into symbols in a 24-step quarter-tone scale. This was done in order to correctly identify tones that were off by half a semitone, which should be considered correct (see section 3.2). This rounding-up to a 24-step scale involves a round-up error of the order of 1/2 of a quarter-tone, or 1/8 of a tone, and so the total tolerance adopted for this analysis was actually 3/8 of a tone, which is not so much if natural vibrato is taken into account.

The last step of the audio analysis consists of grouping up those symbols of the 24-step quarter-tone scale corresponding to a single profile and translate them into pairs (N,P) where N stands for a possible note (such as C#3 or B4) and P is the percentage of time of that profile for which the note could be accepted as N. For instance, an output like the following

```
----------------------------------------
Event Detected:          Intensity=411.783
Start=0.673s  End=1.196s  Duration=0.522s
Possible Notes: D 4 (99%), C#4 (29%)
----------------------------------------
Event Detected:          Intensity=515.923
Start=1.428s  End=1.974s  Duration=0.546s
Possible Notes: C 4 (99%), B 3 (60%)
----------------------------------------
```

states that the first note could be accepted as a D4 for 99% of that utterance's duration, but it could also be accepted as a C#4 for 29% of the time (it might be the case that the note was a little bit flat during attack or decay), whereas the second note could be either C4 or B3 (because F0 values were in between these two notes for 60% of the time). So the output of this analysis can be seen as a probabilistic transcription taking the tolerance of 3/8 of a tone into account.

All recordings were semi-automatically graded by this transcription system. By that we mean that conversion of recordings into span measures has been double-checked by a musically trained person. This was done for two main reasons: (1) to minimize the possibility of automatic transcription errors being transferred to the statistical analysis, with an impact in cognitive results; and (2) to gather extensive subjective evaluation about the transcription system, by applying it to over 800 recorded notes, and verifying that correct notes (according to a musically trained person) were always identified by the transcription system with P>33%.

4.2 Statistical Comparison of Span Results

The output from the previous analysis is a set of numerical measurements for each individual and each test category. For simplicity, these categories were labeled with short names such as 7 and 12 for diatonic and chromatic span measures, and 3 and X for the categories related to difficulty (3 = intervals constrained to at most a major third, and X = no interval constraint). This data was submitted to a Repeated Measure Two-Way ANOVA on the effects of familiarity and difficulty, and post-hoc Newman-Keuls tests when necessary, using Statistica© r5. Each possible comparison between groups of measures that might be statistically different has a corresponding significance level p, and small values of p (typically $p<0.05$) are interpreted as indicating a real difference between groups.

Figure 2 shows the averages and standard errors for the melodic span measures in direct order, or forward melodic spans (FMS), in all four categories.

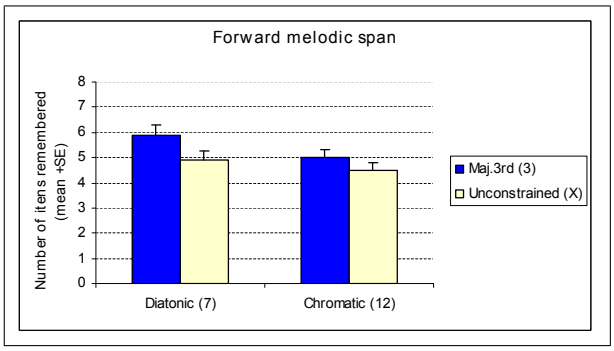


Figure 2. Forward melodic span measures.

The average forward melodic span for category 7_3 (5.9 notes) was significantly higher than the others (FMS(7_X)= 4.9, FMS(12_3)=5.0 and FMS(12_X)=4.5), with a significance level $p<0.023$. We may assume that the smaller number of items combined with a simpler internal structure does in fact ease the memorization task. Pairwise comparisons between the other 3 categories do not show statistically significant differences. This is not equivalent to saying that there are no differences, but simply that the experimental data for this population does not allow such conclusions to be drawn with reasonable confidence. A larger population might improve significance levels, allowing other hypotheses of pairwise comparisons, such as span(12_3)>span(12_X), to be confirmed or refuted.

With a two-way ANOVA we can study the effects of the familiarity (scale) irrespectively of difficulty (constrained or unconstrained melodic leaps), by combining all measures for the diatonic scale (7_3 and 7_X) and statistically comparing this group of measures to the results for the chromatic scale (12_3 and 12_X). This comparison allows us to conclude that average measures for the diatonic scale (FMS(7)=5.4) are significantly higher ($p<0.018$) than measures for the chromatic scale (FMS(12)=4.75). Comparing the two levels of difficulty irrespectively of familiarity leads to a similar conclusion, i.e., average measures for constrained sequences (FMS(3)=5.45) are significantly higher ($p<0.026$) than for unconstrained sequences

(FMS(X)=4.7).

It is interesting to compare these results to the corresponding span measures for numerical sequences that were built after melodic sequences by direct translation. Figure 3 shows the results of these tests. The labels (7), (12), (3) and (X) have been maintained, although in this context they only reflect the amount of allowed numbers (8 or 13) and the allowed differences between adjacent numbers.

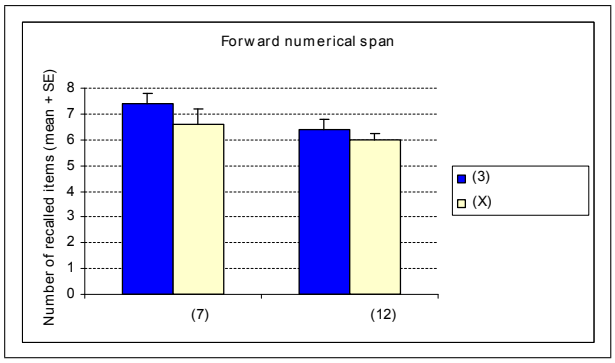


Figure 3. Forward numerical span measures.

Considering all four categories, forward numerical spans are higher than forward melodic spans ($p<0.020$), which means that sequences of numbers are more easily recalled than melodic sequences in the context of this experiment.

Here we also observe the same combined results as before, namely FNS(7) is significantly higher than FNS(12) ($p<0.006$), and FNS(3)is significantly higher than FNS(X) ($p<0.024$). This raises some important questions about the interpretation of melodic span results, which will be addressed in section 4.3.

We now turn to melodic spans in inverse order, or backward melodic spans (BMS). Figure 4 shows averages and standard errors for this experimental data.

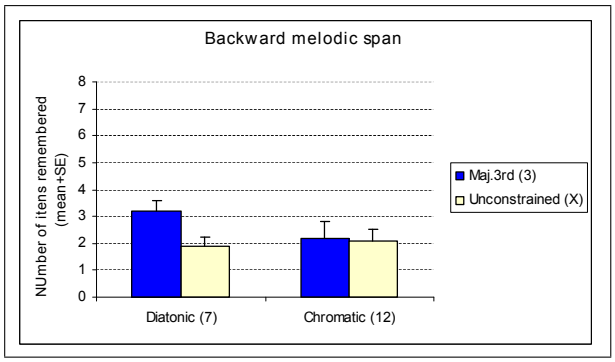


Figure 4. Backward melodic span measures.

The only worthy comparison here is between BMS(3)= 2.7 and BMS(X)=2 which refer to difficulty levels, but the significance level $p=0.094$ is higher than 0.05, which means that the confidence in this comparison is relatively low. This might be confirmed with a larger population.

It should be mentioned that these backward span measures are affected by the presence of several zeros corresponding to subjects who couldn't reproduce any sequences in reverse order (sequences start with 2 distinct tones). This, combined with many other low results (BMS=2) contributes to what is called *floor effect*, which has important consequences for statistical analysis of these data.

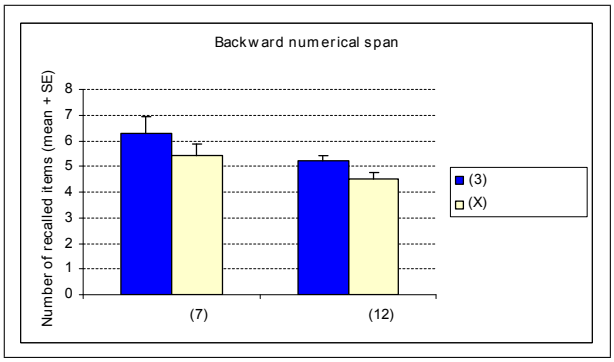


Figure 5. Backward numerical span measures.

Figure 5 shows backward numerical span measures. This data allows the conclusions that BNS(7)=5.85 is higher than BNS(12)=4.85 ($p<0.017$), and that BNS(3)=5.75 is higher than BNS(X)=4.95 ($p<0.019$), or in other words, both the smaller number of symbols and the simplified interval structure of the sequences do in fact help the memorization and mental reversal of sequences of numbers.

It can also be drawn from this data that backward numerical span measures are higher than the corresponding backward melodic spans ($p<0.005$). This means that retrograding melodic sequences is in fact much more difficult than reversing numerical sequences, and the confidence level of this conclusion is high.

The next section focuses on possible cognitive interpretation of the above quantitative and qualitative conclusions.

4.3 Cognitive Aspects

We shall first address the differences and similarities in forward span measures for melodic and numerical sequences. We concluded in section 4.2 that numerical span measures were generally higher than melodic span measures. This could be explained by the many associations that numbers in working memory have with long-term memory knowledge, such as visual and linguistic alternative representations. A similar phenomenon has been observed in individuals with absolute pitch, who resorted to verbal strategies to achieve a higher melodic span [14].

Another interesting comparison is the fact that the restricted contexts (7 and 3) did increase forward span measures with respect to less restricted contexts (12 and X), both with melodic and numerical sequences. This raises the possibility of a single explanation accounting for both phenomena, which might not be an exclusively musical explanation. Items (numbers, pitches) that are close to one another in their respective representation spaces might be more effectively combined into larger chunks (subsequences, motifs) in the working memory, effectively allowing a larger number of items to be stored.

It has been observed that the number of symbols (8 or 13) also affect span measures. This effect might be intuitive in the numerical domain, since some numbers are represented by two digits and might also have a comparatively

larger mental representation. But in the musical domain we have been looking at those categories (7 and 12) as representatives of more or less familiar contexts. It might be argued that a single explanation (number of symbols) would account for both observations. We would counterargue that chromatic sequences with length less than 8 also have less than 8 distinct symbols, so non-diatonic 8-element subsets of a 13-element chromatic scale already appeared in our experiment; the only difference is the fact that these 8-element subsets are not fixed within each category. An experiment might be made using other 8-element fixed subsets of a 13-element chromatic scale to provide a more well-founded comparison.

In backward melodic span measures we observed a floor effect that make it more difficult to draw qualitative conclusions from statistical analysis. It might be the case that chunking of close elements within a musical scale make the process of reversal of a sequence easier. In any case, by comparison with the reversal of numerical sequences, musical retrogradation of unheard melodic sequences appears to be a very difficult task.

It is interesting to notice that backward digit span measures are affected by restricted contexts such as 7 (8 instead of 12 symbols) or 3 (small rather than large intervals). This suggests that chunking of information in working memory is probably more effective in reversing numerical sequences rather than melodic sequences.

It might be wondered about the effect which training would have in both tests. Reversal of numerical sequences does not appear to be a frequently applied task in elementary school, and the same could be said about melodic retrogradation without the aid of a writing pad. Yet the results suggest that dealing with numbers in working memory is naturally easier than dealing with notes, in the sense that our population was not specifically trained for neither of these tasks.

These differences in behavior of backward span measures with respect to forward span measures are made more clear when they are expressed by relative differences or *span indices*, defined as

$$\frac{(\textit{forward span - backward span})}{(\textit{forward span})}.$$

Figures 6 and 7 show these values for melodic indices and numerical indices, respectively.

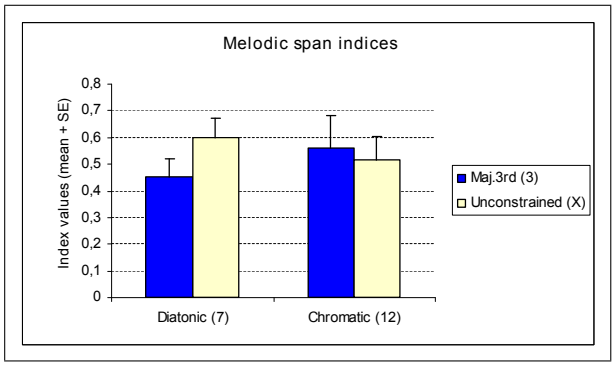


Figure 6. Indices for melodic span measures.

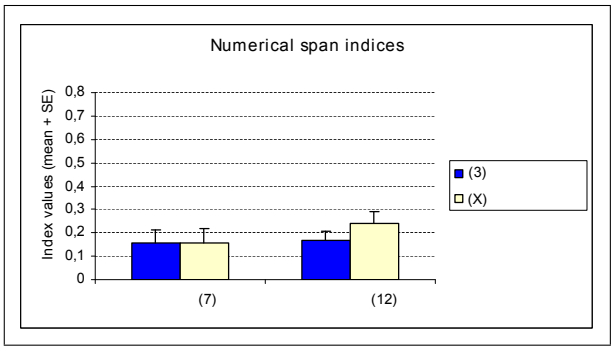


Figure 7. Indices for numerical span measures.

These values reflect the relative difficulty in mentally reversing sequences with respect to simply reproducing them in direct order. Statistical analysis allow the conclusion that numerical indices are higher than melodic indices with a significance level $p<0.0006$.

It is interesting to compare these indices to numerical indices of other languages. For instance, digit span indices for English, Spanish, Hebrew and German are in the range [0.09, . . .,0.26] [10, 15, 16], and this range also includes all four numerical span indices that we obtained.

On the other hand, digit span indices for Mandarin are relatively higher, around 0.48 ± 0.05 according to Hsieh & Tori [17]. This value is closer to what we obtained as melodic span indices. It might be argued that Mandarin is a tonal language, meaning that pitch variation within a phoneme is a component of semantic value, and so even the task of remembering numbers (or reversing them) requires some attention to melodic profile.

These differences suggest that the underlying mechanisms for verbal and tonal processing might be different, as suggested by other authors [7]. Baddeley's working memory model includes separate components for visuospatial and phonological information, but does not distinguish between phonological information with verbal content or purely acoustic information. By observing the differences in numerical and melodic span indices we could consider a subdivision of the phonological loop into two components responsible for verbal and acoustical material, or even the existence of a component for acoustic processing which is separate from the phonological loop.

5. CONCLUSION AND FURTHER RESEARCH

This paper has brought experimental facts about human music cognition, which might be relevant for computational psychoacoustics and for the development of cognitive models for automatic feature extraction. We have studied the impact of familiarity and difficulty in the task of memorizing melodic sequences, by adding simple constraints to the generation of test sequences.

We observed that both familiarity and difficulty (in the sense defined in section 3.1) contribute to higher forward melodic span measure. A similar finding in forward numerical span measures adds to the understanding of the melodic results in two ways: it provides a possible explanation to measure differences related to difficulty as a

consequence of chunking, and it also raises the question of whether the number of symbols alone would be responsible for the observed differences with respect to what we called familiarity.

Observing behavioral differences in backward numerical and melodic span measures, and specially comparing span indices to other well-known experiments, we suggest that the underlying mechanisms for dealing with verbal and acoustic information in working memory are probably not the same, since a similar mechanism operating similarly on both information would not display the observed levels of degradation in backward spans with respect to forward span measures.

The experiment described here can be easily extended and applied to other population groups. Some of the factors that may contribute to relevant findings are: the size of the population, considering other groups such as professional singers or non-singer professional musicians, and also considering other levels of familiarity or difficulty or even other aspects of melodic sequences not contemplated here.

Future work may also combine this type of experiment to neuroimaging techniques to help mapping cognitive subsystems of the working memory model to particular activation areas in the human brain. Some studies that follow this idea are the localization of regions involved in recognition tests with melodic material using PET scans [7], and the localization of areas involved in the subvocal rehearsal strategy of the phonological loop for verbal and melodic material using fMRI [18].

ACKNOWLEDGEMENTS

The authors would like to thank FAPESP, CNPq and AFIP.

6. REFERENCES

[1] Atkinson and Shiffrin, "Human memory: a proposed system and ist control processes," in *K. W. Spence and J. T. Spence (Eds), The Psychology of Learning and Motivation*, Academic Press, 1968.

[2] A. D. Baddeley and G. Hitch, "Working memory," in *The Psychology of Learning and Motivation*, vol. 8, pp. 47–90, 1974.

[3] A. D. Baddeley, "Working memory and language: an overview," in *Journal of Communication Disorders*, vol. 36, pp. 189–208, 2003.

[4] S. Berti, S. Münzer, E. Schröger, and T. Pechmann, "Different interference effects in musicians and a control group," in *Experimental Psychology*, vol. 53(2), pp. 111–116, 2006.

[5] D. Deutsch, "Tones and numbers: Specificity of interference in immediate memory," in *Science*, vol. 168, pp. 1604–1605, 1970.

[6] R. H. Logie and J. Edworthy, "Shared mechanisms in the processing of verbal and musical material," in *Imagery II, edited by D. G. Russell, D. Marks and J. Richardson*, pp. 33–37, 1986.

[7] R. J. Zatorre, A. C. Evans, and E. Meyer, "Neural mechanisms underlying melodic perception and memory for pitch," in *The Journal of Neuroscience*, vol. 14(4), pp. 1908–1919, 1994.

[8] J. T. E. Richardson, "Functional relationship between forward and backward digit repetition and a non-verbal analogue," in *Cortex*, vol. 13, pp. 317–320, 1977.

[9] A. S. C. Thorn and S. E. Gathercole, "Language-specific knowledge and short-term memory in bilingual and non-bilingual children," in *The Quarterly Journal of Experimental Psychology*, vol. 52A(2), pp. 303–324, 1999.

[10] A. Ardilla, "Language representation and working memory with bilinguals," in *Journal of Communication Disorders*, vol. 36, pp. 233–240, 2003.

[11] A. S. C. Thorn, S. E. Gathercole, and C. R. Frankish, "Language familiarity effects in short-term memory: the role of output delay and long-term knowledge," in *The Quarterly Journal of Experimental Psychology*, vol. 55A(4), pp. 1363–1383, 2002.

[12] A. Mitre, M. Queiroz, and R. Faria, "Accurate and efficient fundamental frequency determination from precise partial estimates," in *Proceedings of the 4th AES Brazil Conference*, pp. 113–118, 2006.

[13] M. Desainte-Catherine and S. Marchand, "High precision fourier analysis of sounds using signal derivatives," in *Journal of the Audio Engineering Society*, vol. 48(8), pp. 654–667, 2000.

[14] M. E. Benassi-Werke, "Memória operacional para tons, palavras e pseudopalavras em músicos," in *Anais do SIMCAM4 - IV Simpósio de Cognição e Artes Musicais*, pp. 1–9, 2008.

[15] H. Silver, P. Feldman, W. Bilker, and R. C. Gur, "Working memory deficit as a core neuropsychological dysfunction in schizophrenia," in *American Journal of Psychiatry*, vol. 160(10), pp. 1809–1816, 2003.

[16] T. Merten, P. Green, M. Henry, N. Blaskewitz, and R. Brockhaus, "Analog validation of german-language symptom validity tests and the influence of coaching," in *Archives of Clinical Neuropsychology*, vol. 20, pp. 719–726, 2005.

[17] S.-L. J. Hsieh and C. D. Tori, "Normative data on crosscultural neuropsychological tests, obtained from mandarin-speaking adults across the life span," in *Archives of Clinical Neuropsychology (in press)*, 2007.

[18] S. Koelsh, K. Schulze, D. Sammler, T. Fritz, K. Müller, and O. Gruber, "Functional architecture of verbal and tonal working memory: an fmri study," in *Human Brain Mapping*, vol. 30, pp. 859–873, 2009.

[19] W. L. Berz, "Working memory in music - a theoretical model," in *Music Perception*, vol. 12(3), pp. 353–364, 1995.

ANALYZING GESTURE AND SOUND SIMILARITIES WITH A HMM-BASED DIVERGENCE MEASURE

Baptiste Caramiaux, Frédéric Bevilacqua, Norbert Schnell
UMR STMS IRCAM - CNRS
1, place Igor Stravinsky
75004 Paris, FRANCE
baptiste.caramiaux@ircam.fr

ABSTRACT

In this paper we propose a divergence measure which is applied to the analysis of the relationships between gesture and sound. Technically, the divergence measure is defined based on a Hidden Markov Model (HMM) that is used to model the time profile of sound descriptors. Particularly, we used this divergence to analyze the results of experiments where participants were asked to perform physical gestures while listening to specific sounds. We found that the proposed divergence is able to measure global and local differences in either time alignment or amplitude between gesture and sound descriptors.

1. INTRODUCTION

Our research is concerned with the modelling of the relationships between gesture and sound in music performance. Several authors have recently shown the importance of these relations in the understanding of sound perception, cognitive musical representation and action-oriented meanings ([1], [2], [3]), which constitutes a key issue for expressive virtual instrument design ([4], [5]).

A gesture is described here as a set of movement parameters measured by a motion capture system. In turn, a sound is described as a set of audio descriptors representing musical properties such as audio energy, timbre or pitch. Specifically, our goal is to propose a computational model enabling the measure of the similarities between the gesture parameters and sound descriptors.

Previous works on the quantitative analysis of the gesture-sound relationship often deal with variance-based statistical methods as principal correlation analysis (PCA) ([6]) or canonical correlation analysis (CCA) ([7]). PCA allows for the determination of principal components that models the variation of the gesture parameters. Analyzing these components together with musical features (as tempo or metric) enabled to understand how listeners try to synchronize their movements on music beats ([6], [8]). In [7] the CCA method is used as a selection tool for mapping analysis. In this work, we showed that this method can return the gesture and sound predominant features. However, both variance-based methods suffer from a lack of temporal modeling. Actually, these models assume as stationary both gesture parameters and audio descriptors, in the sense that statistical moments (mean, variance, etc.) do not depend on the ordering of the data. As a matter of fact, these models return a global static similarity measure without considering intrinsic dynamic changes.

To overcome these limitations, it is necessary to model the time profiles of the parameters. A large number of works dealing with time series modelling are based on hidden Markov models. HMM-based methods indeed allow for the temporal modeling of a sequence of incoming events, and have been used in audio speech recognition [9], gesture recognition ([10], [11]) and multimodal audio-visual speech recognition [12]. The common classification task generally considers a sequence as a unit to be classified and returns a decision once completed based on the computation of likelihood values. In [11] the authors present a HMM method designed for continuous modeling of gesture signals, that allows for the real-time assessment of the recognition process. Moreover, this method allows for the use of a single example for the learning procedure.

We propose to use in order to provide a measurement tool in a cross-modal fashion. HMM were already employed in cross-modal contexts : audio speech and video [13], [14]. Here the novelty is to use HMM methods to model relationships between non-verbal sounds and hand gesture of passive listeners. More precisely, we propose here to use this method to further define a statistical distance between two time profiles, typically called a divergence measure (see for instance [15]) in information processing. Specifically, we report here that this HMM-based divergence measure has properties, induced by its underlying Markov process [16], that makes it suitable to study the time evolution of the similarity between gesture parameters and sound descriptors.

This paper is structured as follows. First, we describe the general method and context of this work. Second, we present the theoretical framework of hidden Markov modeling (section 3). In section 4 we detail the divergence measure based on this framework and a specific learning process. Third, we report an experiment and the results that illustrate a possible use of our method (section 5). Finally, we conclude and present future works in section 6.

2. CONTEXT AND GOAL

Consider the following experiment: a participant listens to a specific sound several times, and then proposes a physical gesture that "mimics" the sound. The gesture is then performed (and captured) while the participant listens to the sound. Our general aim is to answer the following question: how can we analyse the gesture in relation to the sound ?

In this experiment, the gestures can be considered as a "response" to a "stimulus", which is actually the sound. In our framework, we will thus consider the sound as the "model" and the gestures as the "observations", as if they were generated by the model.

For each participant's gestures, as illustrated in figure 1, our model should allow us to compute a divergence measure between each gesture and the corresponding sound (or in other words, to quantify similarity/dissimilarity between the gesture and sound). In the next section, we describe the mathematical framework enabling the computation of such a divergence measure. It is based on Hidden Markov Modeling permitting real time musical applications.

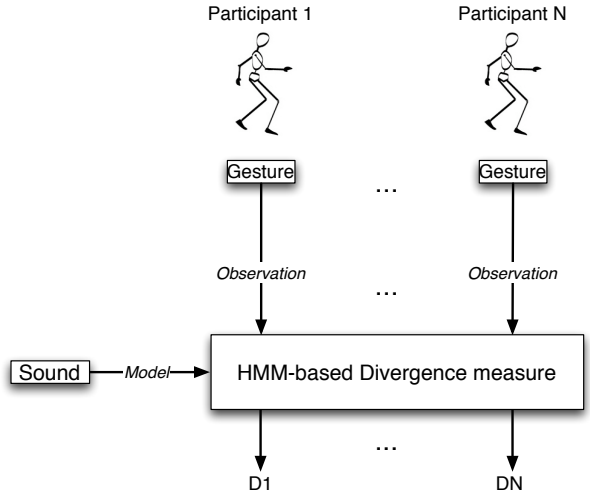


Figure 1. Methodology: Each participant's trials are taken as input and a selected sound is taken as model. We measure the divergence between each trial and the sound.

3. HIDDEN MARKOV MODELING

In this section we briefly report the theoretical HMM framework used to further define the divergence measure in section 4.

3.1 Definition

Hidden Markov modeling can be considered as two statistically dependent families of random sequences O, X ([17], [16], [9]). The first family corresponds to the observations $\{O_t\}_{t\in\mathbb{N}}$ which represent measurements of a natural phenomenon. A single random variable O_t of this stochastic process takes value in a continuous finite dimensional space $\mathcal{O}$ (e.g $\mathbb{R}^p$). The second family of random process is the underlying state process $\{X_n\}_{n\in\mathbb{N}}$. A state process is a first-order time-homogenous Markov chain and takes values in a state space denoted by $\mathcal{X} = \{1, 2, \ldots, N\}$. If we note T the length of O, statistical dependency between the two processes can be written as

$$P(O_1 \ldots O_T | X_1 \ldots X_T) = \prod_{t=1}^{T} P(O_t | X_t) \qquad (1)$$

We define a hidden Markov model as

$$\lambda = (A, B, \pi)$$

Where A is the time-invariant stochastic matrix, or transition matrix, $P(X_{t+1} = j_1 | X_t = j_2), (j_1, j_2) \in \mathcal{X}^2$; B is the time invariant observation distribution $b_j(o) = P(O_t = o | X_t = j), j \in \mathcal{X}$; and π is the initial state probability distribution $P(X_0 = j), j \in \mathcal{X}$. The HMM structure is reported in figure 2.

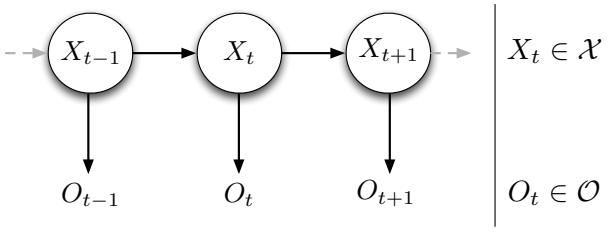


Figure 2. A general schema of HMM. $\{X_t\}_{t\in\mathbb{N}}$ is the model state random process where each state emits an observation O_t with a probability defined by B

In our case, $\{X_0 \ldots X_T\}$ corresponds to an index sequence of audio descriptor samples and $\{O_1 \ldots O_T\}$ a sequence of vector of samples from gesture parameter signals.

3.2 Topology

A and π must be fixed according to a modeling strategy. π describes where in the sequence model we start to decode. A is used to constrain the neighborhood of state j, taken at time t, in which a model state must be taken at the next time step $t + 1$. This data has a great influence on the resulting decoding computation. Let's consider two extreme situations for a forward Markov chain topology as illustrated in figure 3.

In the first case, if current state is j we constrain to look forward until $j + 1$ for the best state emitting O_{t+1} whereas in the second case we allow to look forward until the last state N to find this closest state. Usually, topology is learned from the data to have the most suitable model. Otherwise, we can tuned up the model according to a specific required behavior. For instance, as we work with continuous time series, a forward model will be chosen.

3.3 Learning

Here we present how λ is learned using the approach presented in [11]. The parameters A (transition probability matrix) and π (initial probability) are fixed according to

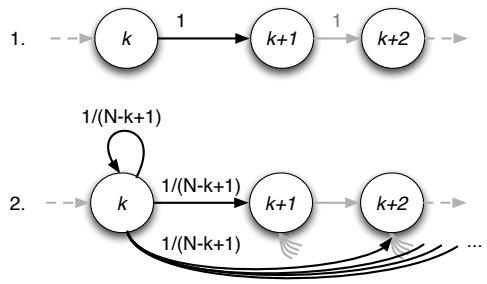


Figure 3. Two extreme cases of topology. First, one step forward is permitted in the state space. Second, each state from the current to the last one can be caught

user's choice of topology. B is such that time invariant observation distributions are gaussians, i.e

$$b_j(o) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-1}{2}\frac{(o-\mu_j)^2}{\sigma^2}\right] \tag{2}$$

Gaussian functions are centered on the model signal samples and the standard deviation σ can be adjusted by the user (see figure 4). In our case, model signal samples are the audio feature samples computed from the chosen sound. A single example, namely the model, is needed for the learning procedure.

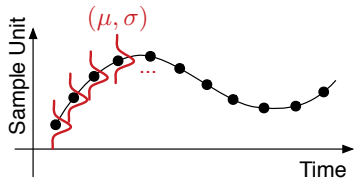


Figure 4. Learning phase. Gaussian functions are centered on the model signal samples and the standard deviation σ is *a priori* defined as a tolerance parameter.

Thereby, rather learning based on training data, the observation probabilities are chosen such that the sound signal is the most likely observation sequence. In this way, we seek for the most likely gesture as the most similar to audio descriptor temporal evolution.

3.4 Decoding

Given an input sequence O and a HMM λ, one of the interesting problems is to compute the probability $P(O|\lambda)$. As mentioned in [9], in practice we usually compute the logarithm of this probability as

$$\log\left[P(O|\lambda)\right] = \sum_{t=1}^{T} \log\left[\sum_{j=1}^{N} \alpha_t(j)\right] \tag{3}$$

Where $\alpha_t(i)$ is called the forward variable and is defined as $\alpha_t(i) = P(O_1 O_2 \ldots O_t, X_t = i|\lambda)$, namely the probability of having the observation sequence $O_1 \ldots O_t$ and the current state i. Also, this variable can be computed recursively providing an incremental method to find the desired probability [9], i.e $\forall j \in [\![1, N]\!]$

$$\begin{aligned} t = 1 \quad & \alpha_1(j) = \pi_j b_j(O_1) \\ t > 1 \quad & \alpha_t(j) = \left(\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right) b_j(O_t) \end{aligned} \tag{4}$$

This forward inference allows for real time applications in which input signal is decoded inductively.

4. DIVERGENCE MEASURE

In this section we define the divergence measure based on the HMM framework and the learning method described in section 3.3. Three main properties of this divergence are proved below: non-negativity; global minimum; non-symmetry.

4.1 Divergence Measure Definition

We consider two uniformly sampled signals: a model $M = \{M_1, \ldots, M_N\}$ and an observation $O = \{O_1, \ldots, O_T\}$. We define here the divergence measure between the observation O and a HMM learned from signal M as in section 3.3, based on decoding presented in section 3.4. We denote $\lambda_M = (A_M, B_M, \pi_M)$ the HMM learned from M. As mentioned in 3.3, we fix A_M and π_M for the divergence independently to M. Observation distributions b_j^M are defined as equation (2) with $\mu_j = M_j$. Hence we have $\lambda_M = (A, B_M, \pi)$. We define the divergence measure as

$$D_{A,\pi}(O||M) = -\log\left[P(O|\lambda_M)\right] \tag{5}$$

In the following, for convenience $D_{A,\pi}$ will be noted D. Divergence measure corresponds to the logarithm of the likelihood of having the sequence of observations O given a model λ_M learned from a signal M. More precisely, $D(O\|M)$ measures the divergence between the input observation and a sequence of model states generating the observations. This sequence respects temporal structure of the model thanks to the underlying Markov chain. The result is a temporal alignment of model states on observations with a probabilistic measure evaluating how the alignment fits the observation sequence in terms of time stretching and amplitude (cf figure 5).

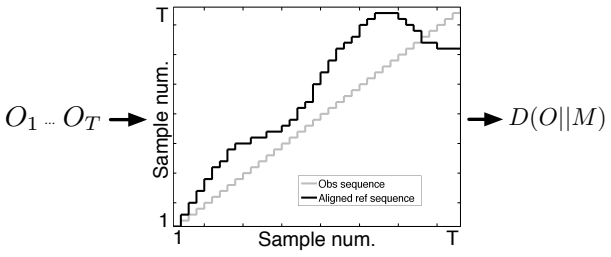


Figure 5. The HMM takes as input the sequence of observations $O_1 \ldots O_t$. A sequence of model states (whose likelihood of emitting observations is maximum) approximates the observations. The quality of modeling is returned and defines $D(O\|M)$.

4.2 Divergence Properties

In this section, we present that divergence measure between observation O and model M defined by (5) satisfies important properties. We refer the reader to the appendix for more details.

Non-negativity Divergence $D(O\|M)$ is always positive. Theoretically, the divergence measure does not have to be finite. Actually, $D(O\|M)$ is finite because signals considered have a finite length ($T, N < +\infty$) and infinite values are theoretically impossible, due to numerical precision. The log of very small values can be either considered as zero or disregarded.

Lower bound. The most important corollary of non-negativity is the existence of a lower bound i.e a global minimum for our divergence measure which varies according to parameters A, π, σ. Moreover, the global minimum is explicit. Depending on A and π, the minimum $D(M\|M)$ is not necessarily zero. Minimum analysis returns how close the HMM learned from M can generate O. In section 5.3 we will show that extremum analysis is pertinent in the analysis of the similarities between a sound and a gesture performed while listening to it.

For brevity, explicit global minimum is not reported here and its analytic formulation will not be explicitly used in the following.

Non-symmetry. The measure is not symmetric. Strategies to symmetrize divergence measures can be found in the literature (see for instance [18] for the well known Kullback-Liebler divergence), but we are interested here in the analysis of the divergence from an observed gesture to a fixed sound model and there is *a priori* no reason why their relation should be symmetric.

4.3 Temporal evolution of the measure

The considered sample-based learning method trains an HMM that closely models the time evolution of the signal. Moreover, from forward decoding we can find at each time t which model state emits the considered observation. Thus, at each time step the model can inform us on the close relation between both signals in terms of time evolution and amplitudes. This aims to an explicit temporal evolution of the divergence measure. Let any truncated observation signals be denoted by $O_{|t} = \{O_1 \dots O_t\}$ and the whole model λ_M. Hence D is defined as a function of time by,

$$D(O_{|t}\|M) = -\sum_{k=1}^{t} \log\left[\sum_{j=1}^{N} \alpha_k(j)\right] \qquad (6)$$

5. EXPERIMENTS

In this section we present an evaluation of the previously defined divergence measure to gesture and sound data. The measure returns an overall coefficient of the similarity between descriptors of both sound and performed gesture. Temporal evolution of this measure allows for the analysis of temporal coherence of both signals. We discuss the results at the end of this section.

5.1 Data Collection

The data was collected on May 2008 in the University of Music in Graz. For the experiment 20 subjects were invited to perform gestures while listening to a sequence of 18 different recorded sound extracts of a duration between 2.05 and 37.53 seconds with a mean of 9.45 seconds. Most of the sound extracts were of short duration. Since the experience was explorative, the sound corpus included a wide variety of sounds: environmental and musical of different styles (classical, rock, contemporary).

For each sound, a subject had to imagine a gesture that he or she performed three times after an arbitrary number of rehearsals. The gestures were performed with a small hand-held device that included markers for a camera-based motion capture system recording its position in Cartesian coordinates. The task was to imagine that the gesture performed with the hand-held device produces the listened sound. A foot-pedal allowed the beginning of the movement to be synchronized with the beginning of the playback of the sound extract in the rehearsal as well as for the recording of the final three performances.

5.2 Data Analysis

We refer the reader to the previously introduced method in figure 1. We first select a sound as a model. This sound is *waves*. It is a sequence of five successive rising and falling ocean's waves at different amplitudes and durations. According to the sound model, we consider the three trials performed by each candidate while listening to it.

The divergence measure parameters are set as follows. The chosen transition matrix corresponding to the Markov chain topology is illustrated in figure 6 (see [11] for further explanations). The initial probability distribution π is such that $\pi_1(O_1) = 1$ and $\forall i \neq 1, \pi_i(O_1) = 0$. The states of the Markov chain are the index of the audio descriptor samples (see section 3.3).

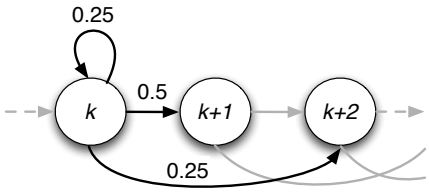


Figure 6. The chosen topology gives the predominant weight to a transition to the next state. An equal weight is given to the self-transition and to the transition above the next state.

The choice of audio description and gesture variables is based on our previous works (cf. [7]). We have shown that the predominant features when participants have performed gestures while listening to a wave sound is the audio loudness and gesture velocity. As we present some results based on the same data, we consider these two unidimensional signals for describing the data.

In the whole set of data captured, some trials had data missing; for others gesture and sound were not synchro-

nized and finally some trials were missing for some participants. A selection is performed based on these criteria. Among the 20 participants, a set of 14 are kept. For all of the 14 participants, we measure the divergence between each trial and the selected sound. Gesture sequence for participant s and trial p is noted $O^{s,p}$, loudness signal is noted M. Figure 7 reports the results.

In the following, we will focus result analysis on four key points.

1. *Divergence Extrema*. Participant performances for which the divergence measure is the lowest and the highest
$$\arg\min_{O^{s,p}} [D(O^{s,p} \| M)]$$
$$\arg\max_{O^{s,p}} [D(O^{s,p} \| M)]$$

2. *Gesture Variability*. Participant performances for which the standard deviation of resulting divergences is low or high.

3. *Temporal Alignment*. Alignment of the model (audio descriptor sample index) onto the incoming observations (gesture parameters): the sequence of states returning the maximum likelihood.

4. *Temporal Evolution*. Evolution of divergence measure for the same selected participant performances as above.
$$D(O^{s,p}_{|t} \| M)$$

5.3 Results and Discussion

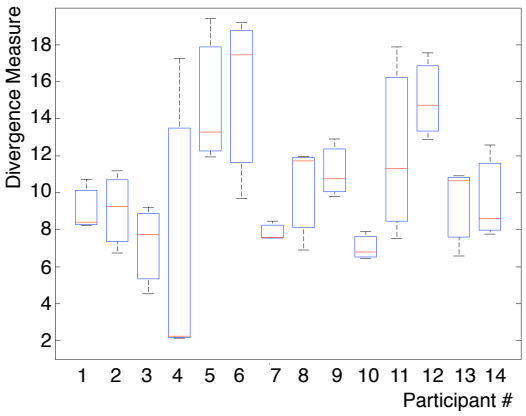


Figure 7. The figure reports statistics on divergence measures between each participant's trial and the sound *waves*. The figure reports each quartile.

Divergence Extrema. Consider first the global minimum and maximum for divergence results obtained on the whole set of data (cf. figure 7). It reveals that participant 4 holds the minimum 2.24 for the second trial. In the same way, participant 5 holds the maximum 19.42 for the second trial. In figure 8, the participant 4's trial minimizing the divergence measure is plotted on the top-left together with the model. On the top-right of figure 8, we report the participant 5's trial maximizing the divergence together with the model. It reveals that participant 4's gesture is more synchronized to the sound and the variations in velocity amplitude fit the best loudness proper variations than participant 5's performance. Actually, participant 4 tends to increase his arm's velocity synchronously with each wave falling. Otherwise, participant 5's gesture performance velocity does not globally correspond to the corresponding loudness variations.

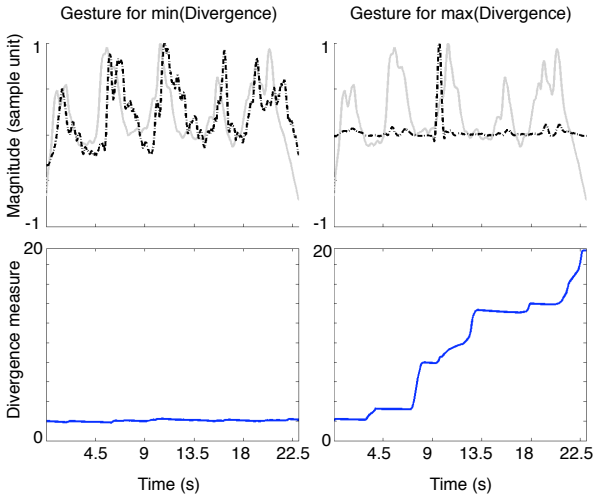


Figure 8. At the top, both gesture velocity signals are plotted in dashed line for both participant 4 (left) and participant 5 (right). The model (*waves* loudness) is also plotted in solid gray line. The bottom is divergence measure at each t between the respective signals above the plot.

Gesture Variability. Illustration of standard deviation between trial divergences in figure 7 reflects the tendency of each participant to perform similar trials in terms of temporality and amplitude. Participant 7 performed very consistent trials compared to participant 4. Divergence medians suggest that a considered participant performed three different gesture performances (e.g. participant 2 or 11) or one really different compared to the remaining two (e.g. participant 4: the first performance is very distinct from the other ones). Figure 9 illustrates this analysis reporting the three trials performed by participants 4 and 7.

In the following, temporal alignment and the resulting temporal evolution of divergence are analyzed on particular examples highlighting how we can interpret the use of such measure for cross-modal analysis.

Temporal Alignment. The divergence measure drastically decreases if both signal amplitude variations differ (see figure 8). A standard correlation measure would behave similarly. The underlying stochastic structure overcomes this limitation by aligning both signals taking into account the ordering of the data. Figure 10 illustrates participant 10's second performance: at the top, original signals (*waves*' loudness and gesture's velocity); at the bottom, the aligned loudness onto the gesture's velocity signal. Even if both signals are not strictly synchronous, the

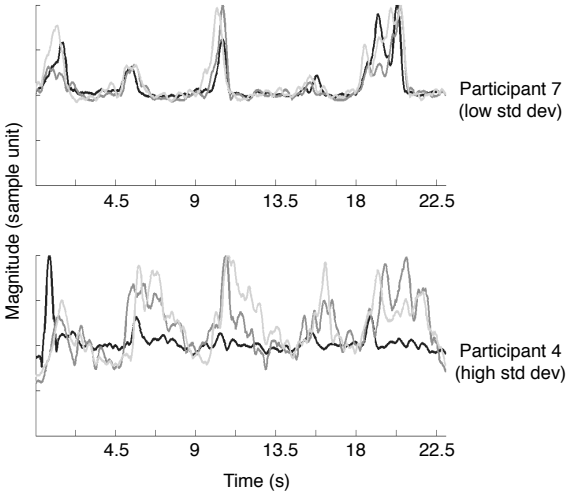


Figure 9. At the top are the trials for which variance in divergence measure is the lowest. Below we plot trials performed by participant 5 and 6 corresponding to the highest variance. Divergence median for participant 5 is roughly the mean (three different trials) of divergence values whereas divergence median for participant 6 is very low (one very different trial from the others)

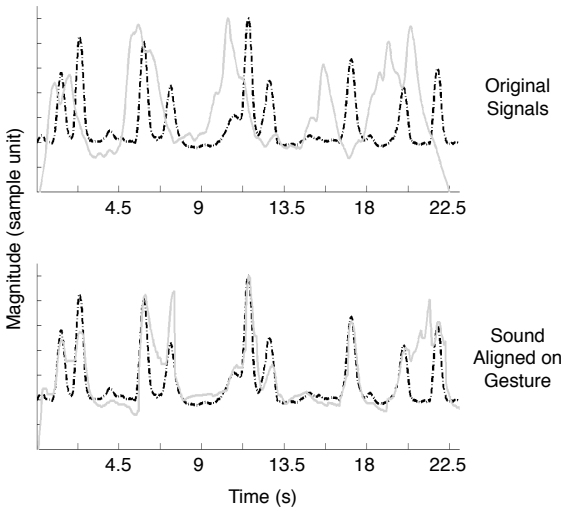


Figure 10. Temporal alignment of loudness onto gesture's velocity. At the top are plotted the original signals : gesture's velocity in dashed line and loudness in solid line. At the bottom, gesture's velocity is unchanged and loudness is aligned onto the velocity signal.

divergence is quite low (6.79). Actually, both signal shapes are globally coherent. The alignment is roughly a time shift of the sound signal resulting from a delayed gesture during the performance. In this example, correlation coefficient before the alignment process would be 0.076 and 0.32 afterwards. Resulting aligned sound could be reconstructed and strategies of reconstruction should be investigated.

Temporal Evolution. As explained in section 4.3, the quality of model state sequence according to observation signal can be measured at each time t. At the bottom of figure 8 are the divergence measures evolving over time for the second trial of participant 4 (left) and the third of participant 5 (right). On the one hand, let's analyze bottom left plot corresponding to participant 4's performance (see figure 11 for a better view of the divergence curve). The first samples of O and M are similar. Incoming observations have a tiny delay and the algorithm realigns both signals. The divergence decreases meaning that amplitudes are close (relatively to σ) and the signals are quite synchronous. Around 2 seconds, the divergence increases: gesture velocity is very low whereas sound loudness is still high. Performer's movement changed of direction involving a decreasing velocity. A peak of divergence informs us at which time a divergence occurs and its magnitude permits the degree of mismatching to be evaluated. In this example, a magnitude of 0.1 represents a small mismatch as illustrated in figure 11 (top part). Thanks to the underlying stochastic structure, the state sequence corrects itself according to the new inputs. Indeed, the divergence measure is then decreasing slowly since the sum over time (from 1 to t, see equation 3) of the log-probabilities induces a memory of the past signals' mismatching. Global shape presents sawtooth-type variations interpreted as local mismatching (peak which magnitude depends on the amplitude difference) and correction (release) (see figure 11).

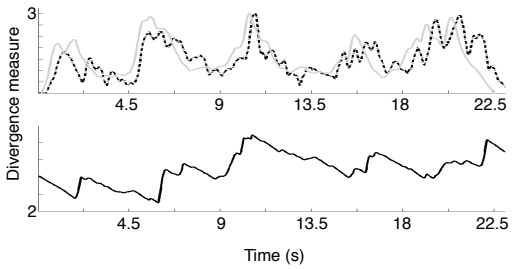


Figure 11. Zoom on divergence measure curve for participant 4. Zooming into this curve illustrates sawtooth-type behavior of the divergence.

Consider now gesture performed by participant 5, shown in the right part of figure 8. The global evolution of the divergence measure is increasing indicating that they globally diverge, contrary to the previous behavior, and its magnitude is higher. The temporal shape shows constant parts (as around 4sec, 9sec, 13.5sec and 18sec). During these intervals, mismatching has less impact because amplitude of both signals is lower. The peaks occur for non-synchronized peaks meaning highly divergent amplitude values. Contrary to the respective bottom-left plot, no decreasing can be seen due to the overall past divergence values that are not good enough to involve a decrease in the divergence: as seen before, the sum propagates past mismatching.

Thereby, two different dynamic behaviors for the diver-

gence measure have been highlighted. Locally mismatching induced a saw shape for $D(O_{|t}\|M)$ whereas globally mismatching induced an ascending temporal curve which can roughly be approximated as piecewise constant. These behaviors give us useful hints to understand dynamic relationships between gesture and the sound which was listened to highlighting relevant parts of the signals where both signals are coherent or really distinct. Unfortunately, the current model does allow the speed of the decrease to be parametrized in the model. Otherwise, since the method considers a global model corresponding to the whole sound signal, it should be interesting to analyze gesture-sound relationship at an intermediate temporal scale between the sample and the global signal. Indeed, changes in gesture control could occur permitting a better fitting between loudness and velocity but the global divergence measure should not take such dynamic changes into account.

6. CONCLUSIONS

In this paper we have presented a divergence measure based on a HMM that is used to model the time profile of sound descriptors. Gestures are considered as observations for the HMM as if they were generated by the model. The divergence measure allows similarity/dissimilarity between the gesture and sound to be quantified. This divergence has the following properties: non-negativity; global minimum; non-symmetry. Experiments on real data have shown that the divergence measure is able to analyze either local or global relationships between physical gesture and the sound which was listened to in terms of time stretching and amplitude variations. Some constraints (changing parameters A, π or σ) could be added in order to reinforce or relax softness of the measure. The novelty is to use HMM methods to model relationships between non-verbal sounds and hand gesture of passive listeners. The use of HMM is motivated by possible real time implementation and interactive applications.

Actually, we are designing a gesture-driven sound selection system whose scenario is as follows. First, we build a sound corpus of distinct audio files with specific dynamic, timbre or melodic characteristics (environmental sounds, musical sounds, speech, etc.). Then we choose an interface allowing physical gesture capturing (e.g. WiiMote). Finally one can perform a gesture and the system will automatically choose the sound for which the divergence measure returns the minimal value. Such application could be useful for game-oriented systems, artistic installations or sound-design software.

7. ACKNOWLEDGMENTS

We would like to thank the COST IC0601 Action on Sonic Interaction Design (SID) for their support in the short-term scientific mission in Graz.

8. REFERENCES

[1] M. Leman, *Embodied Music Cognition and Mediation Technology*. Cambridge, USA: Massachusetts Institute of Technology Press, 2008.

[2] R. I. Godoy, "Gestural-sonorous objects: embodied extensions of schaeffer's conceptual apparatus," *Organised Sound*, vol. 11, no. 2, pp. 149–157, 2006.

[3] F. Varela, E. Thompson, and E. Rosch, *The Embodied Mind: Cognitive Science and Human Experience*. Cambridge, USA: Massachusetts Institute of Technology Press, 1991.

[4] D. Van Nort, "Instrumental listening: sonic gesture as design principle," *Oganised Sound*, vol. 14, pp. 177–187, August 2009.

[5] N. H. Rasamimanana, F. Kaiser, and F. Bevilacqua, "Perspectives on gesture-sound relationships informed from acoustic instrument studies," *Organised Sound*, vol. 14, no. 2, pp. 208 – 216, 2009.

[6] J. MacRitchie, B. Buck, and N. Bailey, "Visualising musical structure through performance gesture," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[7] B. Caramiaux, F. Bevilacqua, and N. Schnell, "Towards a gesture-sound cross-modal analysis," *Lectures Notes in Computer Science, Springer-Verlag*, 2009.

[8] G. Luck and P. Toiviainen, "Ensemble musicians' synchronization with conductors' gestures: An automated feature-extraction analysis," *Music Perception*, vol. 24, no. 2, pp. 189–200, 2006.

[9] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, pp. 257–286, 1984.

[10] A. F. Bobick and A. D. Wilson, "A state-based approach to the representation and recognition of gesture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 12, pp. 1325–1337, 1997.

[11] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana, "Continuous realtime gesture following and recognition," in *Gesture in Embodied Communication and Human-Computer Interaction: Lecture Notes in Computer Science (LNCS)*, Springer Verlag, 2009.

[12] M. Gurban, *Multimodal Feature Extraction and Fusion for Audio-Visual Speech Recognition*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2009.

[13] Y. Li and H.-Y. Shum, "Learning dynamic audio/visual mapping with input-ouput hidden markov models," *IEEE Trans. on Mutlimedia*, vol. 8, no. 3, pp. 542–549, 2006.

[14] M. Sargin, E. Erzin, Y. Yemez, A. Tekalp, A. Erdem, C. Erdem, and M. Özkan, "Prosody-driven head-gesture animation," in *ICASSP'07*, 2007.

[15] I. Csiszár, "Information-type measures of difference of probability distributions and indirect observation," *Studia Scientiarum Mathematicarum Hungarica*, vol. 2, pp. 229–318, 1967.

[16] Y. Ephraim and N. Merhav, "Hidden markov processes," *IEEE Trans. on Info. Theory*, vol. 48, no. 6, pp. 1518–1569, 2002.

[17] J. Silva and S. Narayanan, "Upper bound kullback-leibler divergence for hidden markov models with application as discrimination measure for speech recognition," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2006.

[18] D. H. Johnson and S. Sinanović, "Symmetrizing the kullback-leibler distance," *IEEE Trans. on Info. Theory*, 2001.

A. APPENDIX
DIVERGENCE MEASURE PROPERTIES

Non-negativity.

$$\forall t \in [\![1, T]\!], \sum_{i=1}^{N} \alpha_t(i) = P(O_1 \dots O_t|\lambda_M) \in [0, 1]$$

Hence,

$$D(O\|M) = -\sum_{t=1}^{T} \log \left[\sum_{j=1}^{N} \alpha_t(j) \right] \in [0, +\infty] \quad (7)$$

Lower bound.

Function $b_j^M(o)$ holds a global maximum in $\mathbb{R}^p$ for

$$\forall j \in [\![1, N]\!],\ M_j = \arg\max_x b_j^M(x)$$

For brevity, the whole demonstration is not reported here, but it can be shown that this global maximum aims to a global maximum for $\alpha_t(j)$ leading to a global minimum for the divergence measure $D(O\|M)$ considering any inputs different from the model.

$$\forall O \neq M, D(O\|M) \geqslant D(M\|M) \quad (8)$$

Non-symmetry. From equation (4), let $\alpha_t(j)$ be rewritten as

$$\forall t \geqslant 1,\ \alpha_t(j) = C_{t,j} b_j(O_t)$$

Where $C_{1,j} = \pi_j$ and $C_{t,j} = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}$. From respective expression of $D(O\|M)$ and $D(M\|O)$, we have $\forall t \geqslant 1$,

$$\sum_{j=1}^{N} \frac{C_{t,j}}{\sigma\sqrt{2\pi}} e^{\frac{-(O_t - M_j)^2}{2\sigma^2}} \neq \sum_{j=1}^{N} \frac{C_{t,j}}{\sigma\sqrt{2\pi}} e^{\frac{-(M_t - O_j)^2}{2\sigma^2}}$$

Meaning that the divergence is not symmetric.

$$D(O\|M) \neq D(M\|O) \quad (9)$$

LIMITATIONS IN THE RECOGNITION OF SOUND TRAJECTORIES AS MUSICAL PATTERNS

Blas Payri
Universidad Politécnica de Valencia
bpayri@har.upv.es

ABSTRACT

Spatial movement has been used by composers as a musical parameter (intention), and this paper focus on the reception by the audience of spatial patterns. We present the results of a series of perception experiments where a total of N=118 listeners had to recognize simple rhythm patterns based on the left-right movements of 7 different sound types. The stimuli varied in harmonicity (HNR), temporal intensity variation, spectral distribution, movement continuity and tempo. Listening conditions included stereo loudspeaker open field listening and headphone listening. Results show that globally the recognition is low, considering the simplicity of the pattern recognition task. The factor that most perturbed recognition is the intensity variation, with completely unvarying sounds yielding better results, and this was more important than the listening condition. We conclude that spatial sound movement is not suitable as a composition element for normally complex music, but it can be recognized by untrained listeners using stable sounds and simple patterns.

1. INTRODUCTION

The movement of sound in space is a feature that has been used as soon as allowed by the technology, namely, with the appearance of loudspeakers arrays and their control systems. A famous early piece exploring the use of space is the *Poème électronique* by Varèse created in 1958, with other spatial pieces [13]. Trochimczyk [10] makes a thorough inventory of the multiple uses of spatial sound in electro-acoustic music composition, and cites the intentions of composers when using spatial movements. For instance Xenakis [12] describes the goals when using space: "The composition will thereby be entirely enriched … both in spatial dimension and in movement. The speeds and accelerations of the movement of the sounds will be realized, including logarithmic or Archimedean spirals in time and geometrically … [as well as] ordered or disordered sonorous masses, rolling one against the other like waves". We can see that there was a clear intention of using sound spatial trajectories as an element of the musical language and more recently, the use of space in contemporary music composition and projection has become a widespread musical technique. Composers and music theorists have exposed some ideas to clarify the intention and reception of spatial sound. Sound spatialisation and movement is also a major area in audiovisual sound, particularly with film surround sound systems but also with plain stereo, and manuals provide some hints of do's and do not's when locating and displacing the sounds on screen and around, relating to perception or "in-the-wings" effects in surround sound as theorized by Chion [6]. Videogames and even more sonification systems try to tame spatial movement of sound to convey information and represent spatial data. Sound art works space tends to acquire a more effective role by establishing more direct connections between sounds and the acoustic behaviour of these sounds in a particular environment [4], as oppose to music where the practice of spatialisation remains attached to the idea of sound-source localization and displacement, which is the core topic of the research presented in this paper. The experiments described here try to explore if the use of sound movement can be perceived as a musical element, and if spatial patterns can be recognized.

Sound spatial perception is an area that has received much attention and there is an important set of experimental results [2]. Most experiments deal with source localization rather than movement recognition, and more important, the experiments use very artificial sound stimuli and listening conditions in order to obtain measurable results. Stimuli most often include pure sinewaves and white noise calibrated and stable in time, or chirps that have also very strict conditions on how they are synthesized. These stimuli hardly represent the variety and complexity of the sounds used in actual musical works or in audiovisual productions, and we may not confidently extrapolate the results obtained with calibrated sounds to the real world.

The restrictions on listening conditions are also paramount: most experiments either directly use headphones for hearing the sounds or use anechoic chambers with the listener and the loudspeakers located in very precise positions. For instance, Cheng and Wakefield [5] explain that their "binaural examples have been specifically processed to be listened to over a good pair of headphones. Nonetheless, some of these effects are more successful than others, and we realize that not all listeners may be able to immediately hear the intended spatial effects. Because spatialization effects can be delicate and may vary somewhat from person to person, we suggest listening to each sound example in a quiet environment several times over headphones, closing one's eyes to better concentrate on the sound." These listening conditions are very different from the concert room if we consider music, and also from film theaters or other systems of real-life sound projection.

Finally, the patterns to recognize are very limited: if we deal with static sound localization then there are no movement patterns, and the experiments about moving sound use very simple trajectories, essentially a slow progression in a linear and regular way from one point to another, which does not account for the more complex trajectories usually found in music or audiovisual works.

Our goal is then to study sound spatial trajectories perception, comparing the artificial conditions on stimuli and listening conditions with an approximation to the sound types and listening conditions we may encounter in real concerts, even if we need to simplify greatly. For that matter, we need to create sound stimuli that have a richness in spectrum and temporal evolution that may reflect partially real music sounds, and we use therefore sound synthesis and transformation tools that may apply to electro-acoustic sound generation like the GRMtools.

2. EXPERIMENTAL DESIGN

2.1 Sound spectral and temporal features

An essential question of this research is to study the influence of the spectral and temporal features of the sound on the perception of spatial movement.

We created 7 basic timbres, including:

- 4 synthetic sounds generated with the basic synthesizer of Protools: a pure sine wave, a pure square wave, a pink noise, a mixture of the noise and the square wave
- 3 sounds generated out of recorded sounds (an orchestral soft sound, a voiced vocal sample, and a whispery sound made out of breath) and processed with the Freeze function of the GRMtools, which applies a procedure similar to granular synthesis [7]. These sounds retain the harmonic structure of the original sound although they can last indefinitely in time, and they may have grain and *allure* (temporal amplitude variations) [9]

	Sin.	Sq.	Mix	Nois	Orc	Voc	Brt
Amp SD (dB)	0.00	0.00	0.38	0.70	2.57	3.81	2.04
HNR (dB)	93.7	37.1	4.1	-1.9	7.8	20.4	-3.6
Sp. cent (Hz)	220	478	1390	3488	278	380	4805
Sp. SD (Hz)	1	1032	3195	5346	163	168	3122

Table 1: Amplitude standard deviation in time (dB), harmonicity (harmonics-to-noise ratio, dB), spectral centre of gravity (Hz) and spectral standard deviation (Hz) for the seven basic sounds (sinus, square signal, mix of square and noise, noise, orchestral sound, voiced vocal sound and breath).

We can see in table 1 that the sounds have been generated in such a way that there harmonic samples (sine and square waves, vocal sample), samples including both harmonic and noise components (mix, orchestral sample) and purely noisy sounds (pink noise, breath sample), as can be measured by the Harmonics-to-Noise ratio.

There is also a distribution of temporal amplitude variation as all the synthetic sounds are perfectly stable in time, while all the sampled sounds have variations in time as measured by the Amplitude Standard Deviation.

2.2 Spatial movement patterns

This study focuses on the recognition on spatial movement patterns by listeners, and we decided to choose simple patterns and simple movements, so that the task would be feasible by the listeners without a complex training.

2.2.1 Horizontal plane trajectories

The psychoacoustics of sound spatial perception basically distinguishes three dimensions: the horizontal plane or lateral positions, the elevation or median plane, the distance of the source, and we could add also the source size (Blauert 1997). The human hearing mechanism is more sensitive to sound along the horizontal axis than the vertical axis. Vertical localization depends on subtle filtering of the outer ear and the reflections on the upper part of the body and listeners are not very accurate, and more importantly, perception depends on highly individual corporal differences, which makes listening through headphones very dependent on the actual head and ear shape correspondence between the listener and the recording or filtering device. Distance perception depends on many factors, including the knowledge of the sound spectral characteristics by the listener, the loudness, and the possible reflections.

The human hearing system is well adapted to lateral recognition, thanks to the position of the two ears in the same horizontal plane of the head, which makes it possible and precise both loudspeaker listening and headphone listening. Open field listening uses the differences of the sound arriving to each ear, controlling for the Interaural Time Differences (or more precisely, the interaural phase differences) for lower frequencies, generally below 1500Hz, and the Interaural Level Differences for shorter wavelengths that are affected by the head, usually above 800Hz. Movement types Left-Right movements were applied, using the simple panning feature of Protools. For this exploratory study it was essential to have a robust and easy way to generate the sound stimuli, and we avoided more complex spatial trajectories involving specific software and hardware. Most importantly, L-R movements can be reproduced in any stereophonic system or with any headphones, and we can measure the energy of the sound in the left or right loudspeaker. The goal of the study is to study the perception in close to real conditions, focusing on open field sound reproduction, using real rooms instead of the laboratory conditions of anechoic studios with controlled arrays of loudspeakers and very restrained listeners' positions.

2.2.2 Evolution type of the moving sounds

As we can see in figure 1, the movements were performed from 100% Left to 100% Right.

There were two types of movements: continuous versus discontinuous. In figure 1 we can see the distribution of energy in the left and right channels for a discrete or discontinuous evolution (above) and for a continuous evolution (below). Only the continuous evolution represents a true movement sensation, and the discontinuous evolution has sounds appearing either at one point or the other.

2.2.3 Rhythmic patterns

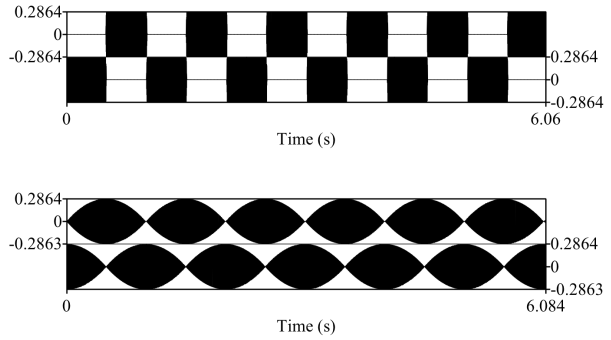


Figure 1: Amplitude/time representation of the Left and Right channels for the long regular rhythm pattern with a continuous movement (below) and a discontinuous *evolution* (above) for a square signal sound.

The task to be performed by the listeners is the recognition of a pattern, understood as a variation in time. Simple rhythmic patterns are easier to explain to listeners, taking in account that we do not have a clearly established vocabulary for spatial movements. The patterns are formed by the Left-Right movement of the sound. Two simple patterns were created: a regular one (see figure 1) and an accelerated one (see figure 2). These simple patterns are easy to understand by untrained listeners and were used for all listeners.

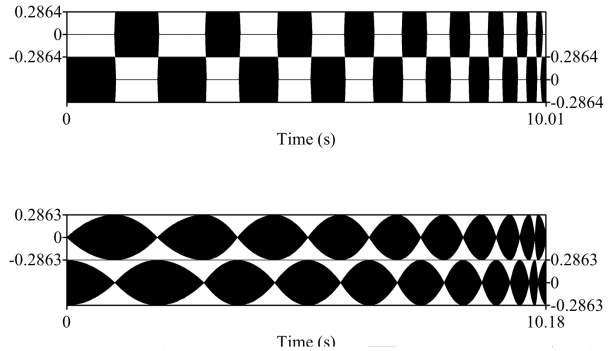


Figure 2: Amplitude/time representation of the Left and Right channels for the long accelerated rhythm pattern with a continuous movement (below) and a discontinuous evolution (above) for a square signal sound.

In addition, we created two more complex patterns, that represented a duple-meter rhythm pattern (figure 3) and a triple meter rhythm pattern (figure 4) that were represented with a musical notation to the musician listeners as can be seen in figure 4.

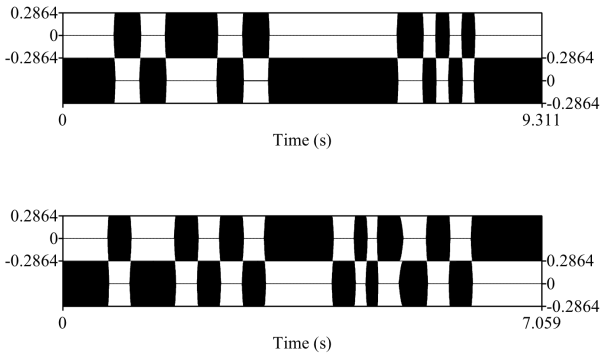


Figure 3: Amplitude/time representation of the Left and Right channels for the long duple-meter (above) and triple meter (below) rhythm patterns with a discontinuous evolution for a square signal sound.

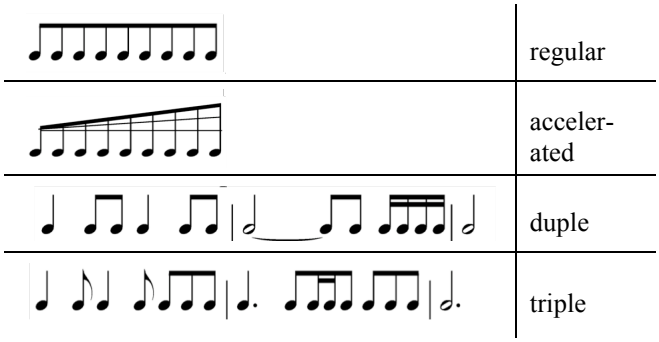


Figure 4: Musical notation of the 4 rhythm patterns as presented to the group of musician listeners.

2.3 Stimuli combination

The sound material is created by the combination of the 7 basic sound timbres, combined with the 4 rhythm patterns, each with either a continuous movement or a discontinuous evolution. To create more diversity, we added a tempo factor, reproducing the rhythm patterns either fast or slow (which represents a duration 1.5 times the duration of the fast version). This represents a total of 112 combinations, or 56 combinations if we consider only the 2 simple rhythm patterns.

2.4 Recognition task

2.4.1 Listeners

A total of 118 listeners participated, including:

- 52 professionally trained musicians, including 47 students of the Master in Music at UPV (courses 2007-08, 2008-09, and 2009-10) and 5 students of the Higher Conservatory of Castellón, that were later eliminated from the computations as their responses were inconsistent.
- 66 students from the film&TV area at UPV (sound students of the Master in Digital Postproduction and students from the Audiovisual Communication degree programme).

2.4.2 Stimuli

The groups of musicians were presented with stimuli representing the 4 rhythmic patterns in figure 4. From the 112 possible combinations, 56 were chosen, representing all the combinations of 7 sound timbres, 4 rhythm patterns, 2 movement types of continuity. Only one tempo alternative was chosen for each combination, alternating faster and slower versions, in such a way that two different sets of stimuli had the alternating fast or slow version of each stimulus. Each group of musicians listened to a combination or the other.

The groups of audiovisual communication students listened to the 56 combinations available with the 2 simple rhythm patterns.

The order of the stimuli presentation was randomized.

2.4.3 Task

Listeners were instructed to listen to the rhythmic patterns, and some examples of the stimuli were played, and then the instructor explained the rhythmic examples with clapping or beating until the differences were clear.

The listeners responded by circling out the right answer among the possible choices in the printed sheet of paper they were handed out. Listeners were also asked to recognize two elements that are out of the scope of this communication: the nature of the movement (continuous versus discontinuous) and whether they heard pitch changes.

2.4.4 Listening conditions

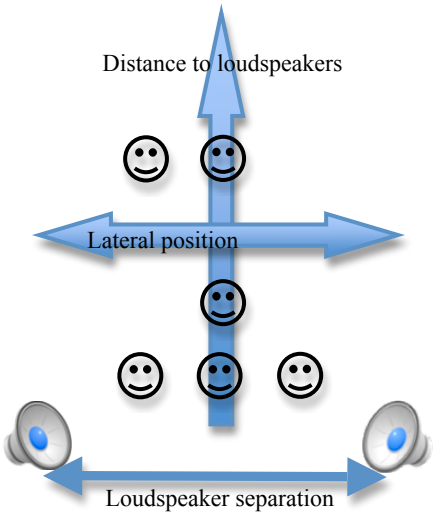


Figure 5: Schema representing the distribution of listeners across the room, and the distances that were recorded.

The main listening setting we wanted to test was open-field listening in an environment close to real life conditions. We used 4 different classrooms, with a low noise level, and in each a set of stereo loudspeakers was used to reproduce the sound. The listeners were distributed in front of the speakers, and we recorded the position of each listener taking in account the distance to the loudspeakers line and the lateral deviation from the loudspeakers center axis as represented in figure 5. We also recorded the loudspeaker separation in each listening session.

To be able to compare the influence of the listening condition, several groups of listeners also heard the sounds through headphones connected to the sound card of a computer were the sounds were played. To control the listening conditions, the headphone listening task was also made in groups, using the same headphones for each listener.

3. RESULTS

3.1 Data

The responses of the listeners were processed to create matrices of 0/1 values, having 1 when the answer of the listener corresponded to the correct pattern (correct recognition) and 0 when the listener did not choose the right pattern, either choosing a wrong pattern or indicating that no pattern was heard.

The answers could then be analyzed as such to compare the results of each listener, or they could be processed as scores representing the mean values for a given sound a condition.

3.2 Factors influencing recognition

3.2.1 Factors under study

We studied the influence of every factor by computing a regression analysis using all the factors and then studying the factors that showed a real influence. We used the following factors:

Sound features as measured with Praat (http://www.fon.hum.uva.nl/praat/): harmonicity (harmonics-to-noise ratio in dB), intensity variation (standard deviation of the intensity in dB), synthetic nature (sampled versus completely synthetic), spectral center of gravity (Hz), spectral bandwidth (spectrum standard deviation in Hz), skewness. As harmonicity and intensity variation had each three distinct value ranges, we added two extra factors that had three possible values (1,2,3) from completely harmonic to completely noisy, and from completely stable to very unstable in intensity.

Movement features: rhythm pattern complexity, movement continuity (continuous versus discreet)

Listening conditions: headphones versus loudspeakers, and with the loudspeaker settings: distance from the listener to loudspeakers, lateral position of the listener from the loudspeaker axis, and loudspeaker separation and quality.

3.2.2 Regression on the factors

Using the factors previously described, we ran a series of regression analyses to explain the recognition level. We can see the factors in order of importance for all the listening conditions (figure 6) and for the loudspeaker condition which is the main condition under study (figure 7): in these figures we use the Beta value from the regression analysis, which is close in nature to a correlation factor.

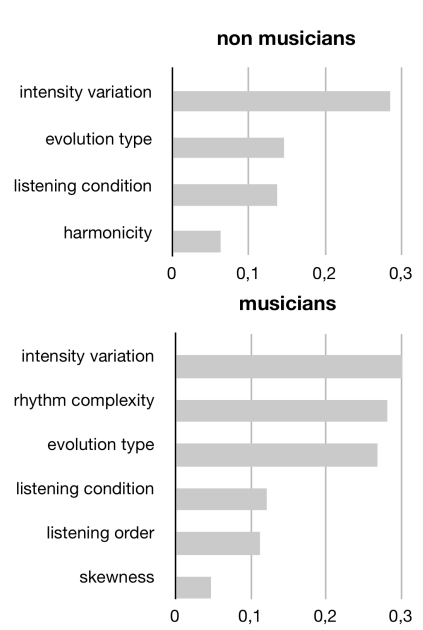


Figure 6: Absolute Beta value as extracted from the regression computation using as a variable the pattern recognition scores for the non-musician group (above) and the musician group (below). Only significant factors are displayed.

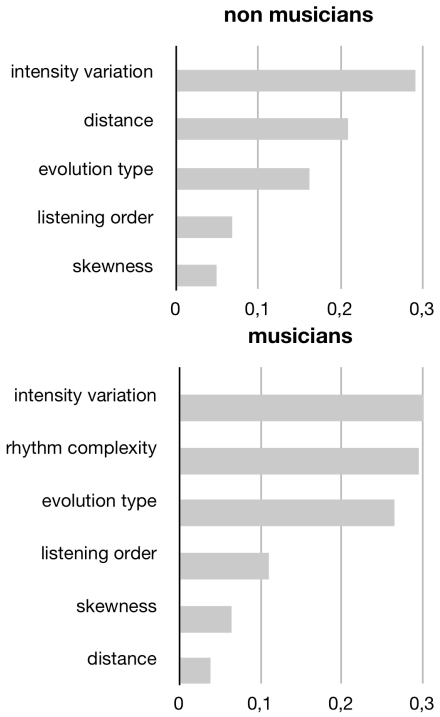


Figure 7. Loudspeaker Listening Condition: Absolute Beta value as extracted from the regression computation using as a variable the pattern recognition scores for the non-musician group (above) and the musician group (below). Only significant factors are displayed.

A surprising result is that the listening condition is relevant in each case, but is not the factor that explains most of the results, as seen in figure 6. In particular, the variations of intensity and evolution type come first both for musicians and non musicians, and the fact that sounds are listened through loudspeakers or headphones comes as a third or fourth factor, and even if the loudspeaker condition did involve a lot of different listening positions that were not optimal. We analyze later the influence of listening conditions and movement type, but the factor that is from far the most influential in the four regression analyses is the intensity variation, that is to say, the sound features are more relevant than the movement features, and those are in turn more important than the listening conditions.

3.3 Influence of the sound features

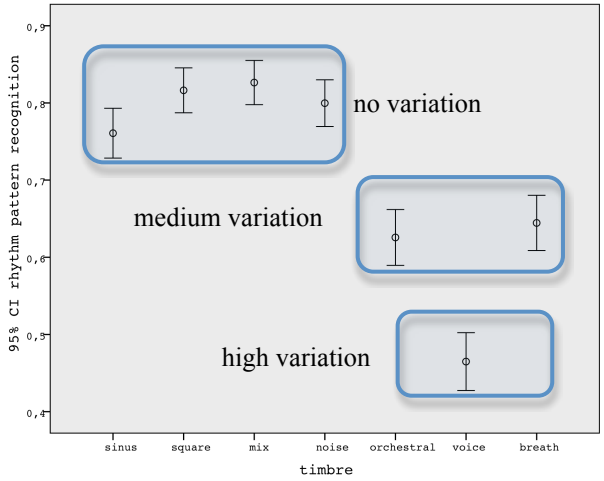


Figure 8: Mean scores (with confidence intervals) of the pattern recognition for each stimuli timbre type. The scores are joined into three groups, labeled according to the temporal intensity variation of the stimuli.

The first analysis concerned the influence of each of the 7 basic sounds we used in the recognition. As can be seen in figure 8, 3 groups clearly appeared, and they corresponded to the level of intensity variation in time of the basic sound. This is the main factor from the sound features, as the harmonicity or other components did not explain the difference in the recognition scores. We retrieve therefore the regression analysis results showing intensity variation as the most important factor.

3.4 Influence of listening conditions

We analyzed the pattern recognition scores depending on the different factors of the sound stimuli and the listening conditions.

We can see in figure 9 that the different basic sound stimuli used have a significant influence on the recognition of spatial movements, but the factor that seems to be really influential is the degree of the temporal variations of the intensity of each sound stimulus, as all the synthetic sounds are completely stable in time and yield the highest recognition scores, where as the sampled sounds have lower scores, and in particular the voice stimulus, which has a higher intensity variability and, correspondingly, a lower recognition score.

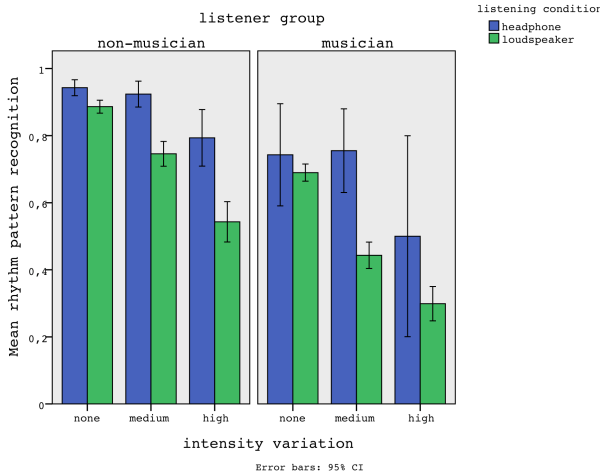


Figure 9: Mean recognition score for the rhythm patterns depending on the group of listeners, the listening conditions and the degree of intensity variation. The scale spans from 1 (100% recognition) down to 0 (0% recognition) and the error bars represent the 95% CI.

3.5 Influence of movement features

As we can see in figure 10, there is a similar pattern in the influence of intensity variation as previously: the recognition ratio drops linearly with the degree of intensity variation, and the other factors like listening condition, movement continuity or pattern complexity may reduce the recognition but as a constant factor to the intensity variation influence.

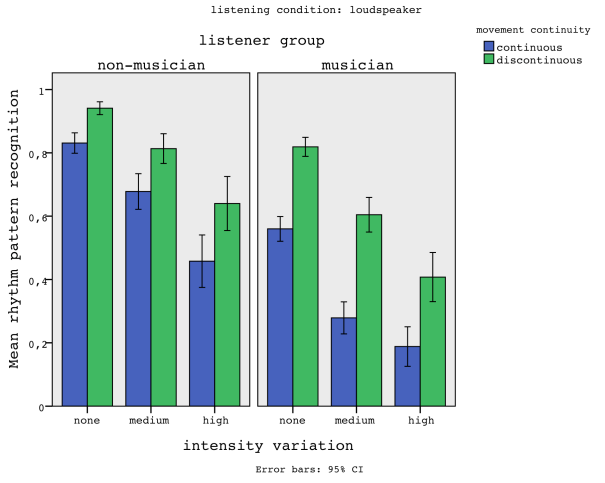


Figure 10: Mean recognition score for the rhythm patterns considering only the loudspeaker listening: results are grouped by movement continuity type and the degree of sound intensity variation. The scale spans from 1 (100% recognition) down to 0 (0% recognition) and the error bars represent the 95% CI.

3.6 Influence of pattern complexity

As displayed in figure 10, movement continuity has a stronger influence on the groups listening to more complex patterns (musician group) and to analyze the exact influence of rhythmic pattern complexity, we performed a series of Chi-square tests on the recognition frequency depending on the sound types. The results show that for the audiovisual group (simple rhythmic patterns) there is a significantly lower recognition rate for the accelerated pattern, which is more complex than the regular pattern. For the musician listeners, two groups clearly appear: the complex patterns (duple and triple metered patterns) have a significant lower recognition rate (p<0,001) than the simple patterns (regular and accelerated).

Figure 11 displays the recognition scores for the different rhythm patterns, with the significant differences circled out.

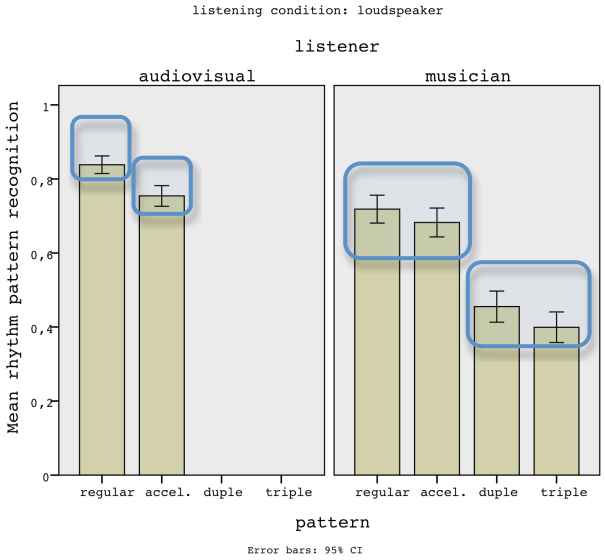


Figure 11: Mean recognition score for the rhythm patterns considering only the loudspeaker listening: results are grouped by pattern type. The scale spans from 1 (100% recognition) down to 0 (0% recognition) and the error bars represent the 95% CI. Results are grouped according to significant differences (chi-square test).

4. CONCLUSIONS

The most salient result of our experiments is that temporal variability is the most influential factor from the parameters we have introduced. This is essential, as either in audiovisual sound (a car moving, people walking…) or in music (electro-acoustic or mixed instrumental), sounds do vary in time, and we never encountered the completely stable synthetic sounds that are used in most experiments of sound spatial perception. For music purposes, most experiments are missing the crucial point by eliminating temporal variability.

The recognition rate is very low for the conditions that are closer to real music or audiovisual sound, i.e. loudspeaker listening of sounds with time variability that evolve continuously, recognition rate drops below 30% (see figure 10), even if we have only 4 possible patterns that are known before hand, and that they are heard alone, without coinciding sounds as we would have in a music or audiovisual work. Considering that we have an oversimplified listening task compared to real conditions, the recognition rate lets us conclude that spatial trajectories

are not suited as a music language parameter with patterns that may be recognized by the audience.

Another result is that musical training does not improve spatial trajectories recognition, maybe considering that this is not a usual musical task. In our case, non-musicians had even better results than musicians, due to the simpler sound stimuli they had to rate. A broad study on the reception of Western music features showed that some musical capacities are acquired through exposure to music without the help of explicit training. These capacities reach such a degree of sophistication that they enable untrained listeners to respond to music as ‘‘musically experienced listeners’’ do [1]. These results are extended to an area more close to rhythm, perception of timing, where judgments are not so much influenced by expertise levels, but by exposure to a certain musical idiom. “As such, the current study provides evidence for the idea that some musical capabilities are acquired through mere exposure to music, and that these abilities are more likely enhanced by active listening (exposure) than by formal musical training (expertise)”[8]. Also, we can see that for the most restrictive task, that is, the synthetic stable stimuli, with simple patterns, the recognition can be superior to 90% and even close to 100% when the sounds do not evolve continuously. There is a clear rhythmic pattern in this case, but this would be achieved more simply with any amplitude variation.

Sound spectral features (harmonics-to-noise ratio, pitch, spectral width) play a residual role compared with temporal intensity variations, whereas experiments that use only stable sounds tend to describe the influence of these features as the most significant [3].

Finally, the listening condition had a lower influence than expected: there was a significant improvement in recognition with headphones as opposed to loudspeakers, but this difference was less important than the influence of sound temporal features. Furthermore, the position of the listener in the room or the separation of the loudspeakers did not have a clear influence on recognition, therefore, when dealing with relatively complex sounds, the composer or sound designer can guess the recognition of the sound movement patterns by the audience, even if a particular listener is not located in the perfect spot (in stereo or surround configurations). That is to say, provided that the composer or sound-designer really pay attention to what can be heard and not heard in spatial movement.

More importantly for musical applications, when listeners were informally asked about their subjective evaluation, they predominantly explained that, even if they could mentally recognize some rhythmic patterns, it was an abstract mental effort of trying to match the spatial trajectory of a given sound with one of the rhythmic patterns, and it was very different from the feeling of recognizing rhythm of changing notes that will emerge as an intuitive musical pattern. “The ability to analyze surface patterns of pitch, attack, duration, timbre in a refined way is probably less important than the ability to integrate all of these features in a structured whole. Without an integrative stage of processing, this perceptual ability, as refined as it may be, would not have any strong implication for musical experience.” [8]

Spatial movement patterns should therefore be restricted to sonification tasks where the sound is perfectly stable in time, and the patterns are very simple, possibly imitating feasible sound trajectories found in nature like the regular pendulum movement or the acceleration that can occur with a bouncing object.

5. ACKNOWLEDGMENTS

This research was partially supported by the project PAID-06-07-3301 funded by Universidad Politécnica de Valencia, and the project GVPRE/2008/377 funded by Generalitat Valenciana, Conselleria d’Educació.

6. REFERENCES

[1] E. Bigand and B. Poulin-Charronnat:. “Are we ‘experienced listeners’? A review of the musical capacities that do not depend on formal musical training.” *Cognition*, Vol. 100, pp.100–130, 2006.

[2] J. Blauert: *Spatial Hearing, the Psychophysics of Human Sound Localization*, MIT press, Cambridge MA, USA, 1997.

[3] D.S. Brungart and B.D. Simpson: “Effects of temporal fine structure on the localization of broadband sounds: potential implications for the design of spatial audio displays,” in *Proc. 14th Intl. Conf. Auditory Display*, Paris, France, June 24-27, 2008.

[4] L. Campesato: “A Metamorphosis of the Muses: Referential and contextual aspects in sound art,” *Organised Sound,* Vol. 14, No. 1, pp. 27–37, 2009.

[5] C.I. Cheng and G.H. Wakefield: “Moving Sound Source Synthesis for Binaural Electroacoustic Music Using Interpolated Head-Related Transfer Functions (HRTFs),” *Computer Music Journal,* Vol. 25, No. 4, pp. 57–80, 2001.

[6] M. Chion: *Audio-Vision: Sound on Screen*, Columbia UP, New York, 1994.

[7] *GRMtools. Freeze specification* (retrieved april 2010) http://www.grmtools.org/qt/files/Freeze.html

[8] H. Honing and O. Ladinig: “Exposure Influences Expressive Timing Judgments in Music”, *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 35, No 1, pp. 281–288, 2009.

[9] P. Schaeffer: *Traité des objets musicaux, 2nd edn.*, Editions du Seuil, Paris, France, 1977.

[10] M. Trochimczyk: “From Circles to Nets: On the Signification of Spatial Sound Imagery in New Music,” *Computer Music Journal*, Vol. 25, No. 4, pp. 39–56, 2001.

[11] I. Xenakis: *Formalized music*, Indiana University Press, Bloomington, USA, 1971.

[12] V. Zouhar, R. Lorenz, T. Musil, J. Zmölnig and R. Höldrich: “Hearing Varèse’s Poème électronique inside a virtual Philips Pavilion,” in *Proc. Intl. Conf. Auditory Display (ICAD 05)*, Limerick, Ireland, 2005.

Examining the role of context in the recognition of walking sounds

Luca Turchet
Medialogy Department
Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup, DK
tur@media.aau.dk

Rolf Nordahl
Medialogy Department
Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup, DK
rn@media.aau.dk

Stefania Serafin
Medialogy Department
Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup, DK
sts@media.aau.dk

ABSTRACT

In this paper, we present an experiment whose goal was to recognize the role of contextual information in the recognition of environmental sounds.

Forty three subjects participated to a between-subjects experiment where they were asked to walk on a limited area in a laboratory, while the illusion of walking on different surfaces was simulated, with and without an accompanying soundscape. Results show that, in some conditions, adding a soundscape significantly improves surfaces' recognition.

1. INTRODUCTION

When exploring a place by walking, at least two categories of sounds can be identified: the persons own footsteps and the surrounding soundscape. In the movie industry, footsteps sounds represent important elements. Chion writes of footstep sounds as being rich in what he refers to as *materializing sound indices* – those features that can lend concreteness and materiality to what is on-screen, or contrarily, make it seem abstracted and unreal [1]. Studies on soundscape originated with the work of R. Murray Schafer [2]. Among other ideas, Schafer proposed soundwalks as empirical methods for identifying a soundscape for a specific location. In a soundwalk people are supposed to move in a specific location, noticing all the environmental sounds heard. Schafer claimed that each place has a soundmark, i.e., sounds which one identifies a place with. The idea of experiencing a place by listening has been recently further developed by Blesser and Salter [3]. By synthesizing technical, aesthetical and humanistic considerations, the authors describe the field of aural architecture and its importance in everyday life.

In the field of virtual reality, studies have recently shown how the addition of auditory cues could lead to measurable enhancement in the feeling of presence. Results are available on sound delivery methods [4, 5] or sound quality [6, 5]. Recently, the role of self-sound to enhance sense of presence in virtual environments has been investigated. By combining different kinds of auditory feedback consisting of interactive footsteps sounds created by ego-motion with static soundscapes, it was shown how motion in virtual reality is significantly enhanced when moving sound sources and ego-motion are rendered [7].

In [8, 9, 10], a system to synthesize in real-time the sound of footsteps on different materials was presented. The system was composed of a set of four contact microphones, a multichannel soundcard, a set of headphones and a laptop. The microphones detected real footsteps sounds from users, from which the ground reaction force (GRF) was estimated. Such GRF was used to control a sound synthesis engine based on physical models.

This interactive system was evaluated in a between-subjects experiment, where it was compared to a recognition task including recorded and synthesized offline sounds. Results showed that subjects were able to recognize most of the synthesized surfaces with high accuracy. Similar accuracy was also noticed in the recognition of real recorded footsteps sounds, which was an indication of the success of the proposed algorithms and their control [9].

In this paper, we are interested in understanding whether the addition of a soundscape enhances the recognition of the simulated surfaces. Our hypothesis is that context plays an important role in the recognition of the material a person is stepping upon. In order to test such hypothesis, we designed different soundscapes, described in the following section.

2. SOUNDSCAPE DESIGN

The soundscapes of the following environments were built:

1. A beach and seaside during the summer
2. A courtyard of a farm in the countryside
3. A ski slope
4. A forest
5. A park during the fall

Such soundscapes were designed according to the indications given by subjects answering to a questionnaire. Precisely, ten subjects, chosen among those not performing the experiment, were asked to imagine which sounds could occur in the above mentioned environments.

Subjects were asked the following question: "Imagine that you are right now in a forest: which sounds do you think you would hear?" In this particular environment,

subjects indicated sounds like trees, birds, different animals. Among the answers provided, we chose those which were stated by more than one subject, and collected a corresponding sound material using appropriate recordings of real sounds.

Such sounds were chosen among those available both on the Hollywood Edge sound effects library [1] and on the Freesound.org website. [2]
The chosen sounds were opportunely edited and assembled using the sound editor Adobe Audition 3. Soundscapes were designed with the goal of providing a clear idea of the designed environment already from the first seconds.

3. EXPERIMENT

We conducted an experiment whose goal was to investigate the ability of subjects to recognize the different walking sounds they were exposed to in three conditions: without soundscapes, with coherent soundscapes and with incoherent soundscapes.
The footsteps sounds provided during the three conditions were synthesized sounds generated in real time while subjects were walking using the interactive system described in the previous section. The soundscapes were audio files played in background independently from the subjects movements. The volumes of both footsteps and soundscapes were set by empirical investigation.

One of our hypotheses was that the recognition would have improved using coherent soundscapes rather than the conditions with no soundscapes and with incoherent soundscapes. Similarly we hypothesized higher evaluations in terms of realism and quality in presence of coherent soundscapes.

3.1 Methods

A between-subject experiment with the following three conditions was conducted:

1. Condition 1: footsteps sounds without soundscapes.
2. Condition 2: footsteps sounds with coherent soundscapes.
3. Condition 3: footsteps sounds with incoherent soundscapes.

Participants were exposed to 10 trials in conditions 1 and 2, and 12 trials in condition 3.

During conditions 1 and 2, 5 stimuli were presented twice in randomized order. The stimuli in condition 1 consisted of footsteps sounds on the following surfaces: beach sand, gravel, snow (in particular deep snow), forest underbrush (a forest floor composed by dirt, leaves and branches breaking), dry leaves. In condition 2 the stimuli consisted of the same footsteps sounds provided in condition 1 with in addition the corresponding coherent soundscape mentioned in section 2.

[1] www.hollywoodedge.com/
[2] www.freesound.org/

During condition 3, 6 stimuli were presented twice in randomized order. The stimuli consisted of footsteps sounds on the surfaces beach sand, snow, forest underbrush, with in addition an incoherent soundscape. As an example in presence of the footstep sound on beach sand the provided soundscapes corresponded to those of footstep sounds on snow (i.e., the ski slope) and on forest underbrush (i.e., the forest environment).

3.1.1 Participants

Forty three participants were divided in three groups to perform the three conditions in a between-subjects experiments (n = 15, n = 15 and n = 13 respectively). The three groups were composed respectively of 11 men and 4 women, aged between 21 and 28 (mean = 23.67, standard deviation = 2.12), 8 men and 7 women, aged between 19 and 38 (mean = 24.67, standard deviation = 5.97), and 6 men and 7 women, aged between 21 and 30 (mean = 24, standard deviation = 3.1). All participants reported normal hearing conditions. All participants were naive with respect to the experimental setup and to the purpose of the experiment.

During the experiment the shoes used by subjects were sneakers, trainers, boots and other kinds of shoes with rubber soil.
The participants took on average about 11, 13 and 16 minutes for conditions 1, 2 and 3 respectively.

3.1.2 Setup

The experiment was carried out in an acoustically isolated laboratory where the setup was installed (see Fig. 2). Participants were asked to walk inside an area delimited by four microphones placed in a square configuration on a medium density fiberboard (MDF). [3] Specifically, we used four Shure BETA 91, [4] high performance condenser microphones with a tailored frequency response designed specifically for kick drums and other bass instruments. The microphones' features made them a good candidate for the purpose of capturing footsteps sounds.
The MDF was adopted in place of the carpeted floor of the laboratory in order to improve the quality of the input signal.

The floor microphones were connected to a soundcard, [5] which in turn was connected to a laptop running the sound synthesis engine. Finally the synthesized sounds, as well as the soundscapes, were provided to the user by means of a set of headphones. [6]

3.1.3 Task

During the experiment the participants were asked to wear a pair of headphones and to walk on the MDF in the area delimited by the microphones. They were given a list of different surfaces to be held in one hand, presented as non-forced alternate choice. Such list included a range of materials wider than those presented in experiment. During

[3] 2.5 x 2 m in size and 1 cm thick.
[4] http://www.shure.com/
[5] We used the Fireface 800 sound card, http://www.rme-audio.com/english/firewire/ff800.htm.
[6] Beyerdynamic DT-770, http://www.beyerdynamic.de/

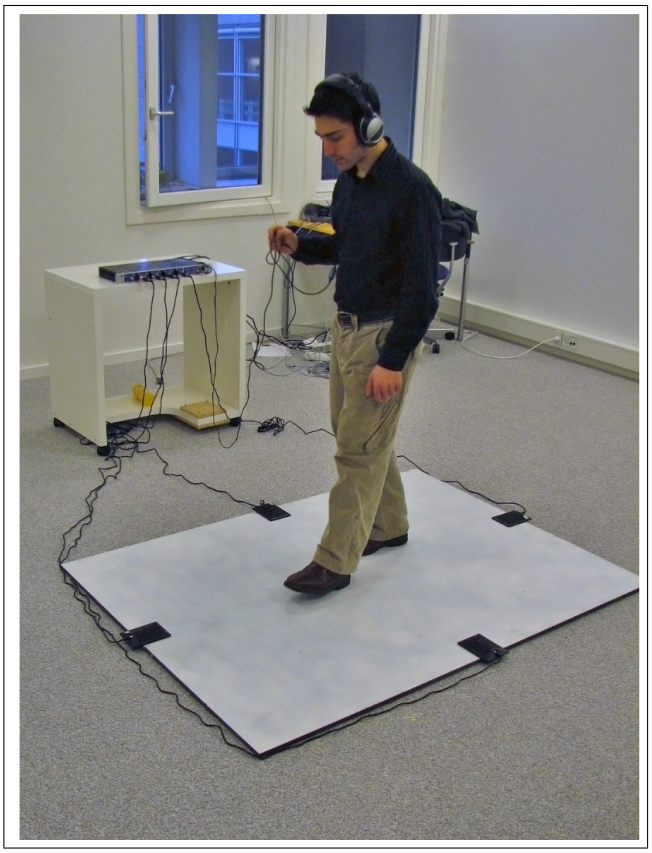


Figure 2. A subject using the interactive footsteps synthesizer. The four contact microphones are clearly noticeable.

the act of walking they listened simultaneously to footsteps sounds on a different surface according to the stimuli presented. The task, common to the three conditions, consisted of answering by voice the following three questions after the presentation of the stimulus:

1. Which surface do you think you are walking on? For each stimulus choose an answer in the following list: 1) beach sand, 2) gravel, 3) dirt, 4) snow, 5) high grass, 6) forest underbrush, 7) dry leaves, 8) wood, 9) creaking wood, 10) metal, 11) carpet, 12) concrete, 13) frozen snow, 14) puddles, 15) water, 16) I don't know.

2. How close to real life is the sound in comparison with the surface you think it is? Evaluate the degree of realism on a scale from 1 to 7 (1 = low realism, 7 = high realism).

3. Evaluate the quality of the sound on a scale from 1 to 7 (1 = low quality, 7 = high quality).

In condition 2 and 3, participants were also asked to recognize what was the environment they were walking.
They were informed that they could choose the same material more than once and that they were not forced to choose all the materials in the list. In addition they could use the interactive system as much as they wanted before giving an answer. When passed to the next stimulus they could not change the answer to the previous stimuli.
At the conclusion of the experiment, participants were also given the opportunity to leave an open comment on their experience interacting with the system.

3.2 Results

The collected answers were analyzed and compared between the three conditions. Results are shown in tables 1, 2 and 3.

The first noticeable element emerging from the three tables is that the use of the interactive system in the condition of coherent soundscapes gave rise to a better recognition of the surfaces and a higher evaluation of realism and quality of the proposed sounds, rather than the conditions with no soundscapes and with incoherent soundscapes. Concerning the percentages of correct answers, they are higher for condition 2 compared to condition 1, for each surface, and the analysis by means of a chi-square test reveals that such differences are statistically significant for beach sand ($p = 0.005515$) and forest underbrush ($p = 0.01904$).
It is particularly interesting to notice that overall adding a soundscape enhances the recognition factor, and this is especially noticeable for those situations where the recognition was rather low without a soundscape.

Similarly the percentages of correct answers are higher for condition 2 compared to condition 3, for each surface, in particular the differences are statistically significant for beach sand ($p = 6.232e-07$), snow ($p = 0.01439$) and forest underbrush ($p = 0.001637$).
Furthermore, the percentages of correct answers are higher for condition 1 compared to condition 3, for each surface, but the differences are not statistically significant.

The analysis of the wrong answers reveals that in all the experiments none of the presented aggregate surfaces was recognized as a solid surface. This means that all subjects were able to identify at least the nature of the surface, which was an expected feature of the simulations. An observation from the subjects performing the experiment was that many subjects perceived the simulated sounds as very similar, and therefore hard to recognize and distinguish from the list provided.

It is interesting to examine what happens when subjects are exposed to soundscapes which are incoherent, as shown in Table 3. In this situation, we consider as correct the answer provided when subjects recognize the surface they are walking on, and not the soundscape. As it can be noticed, the percentage of correct answers is quite low. As expected, adding an incoherent soundscape creates a stronger context which often confuses the subjects. This can be observed, for example, in the case of beach sand footsteps which were rendered together with a forest soundscape and a ski slope soundscape. The recognition rate is higher in the first case than in the second, where several subjects confused sand with snow. The subjects' answers for the three conditions are outlined in the confusion matrices shown in Table 4, 5, 6 respectively. Such matrices show information concerning actual classifications performed by the subjects. From the matrices, it can be noticed how the subjects' recognition varies from condi-

tion 1 to condition 2. As an example, the first row of the matrix illustrated the number of subjects which recognized the beach sand surface, with (Table 4) and without (Table 5) a soundscape. The role of the soundscape to enhance the recognition is clearly noticeable.

Table 6 illustrates the confusion matrix for condition 3, i.e., when incoherent soundscapes are presented to the subjects. In this situation, it is clearly noticeable how the nature of the soundscape plays an important role. Moreover, it can be noticed how the incoherent soundscape is in most situation predominant, in the sense that subjects tend to judge the surface they are stepping upon more listening to the soundscape than listening to the actual surface. On the other hand, even if subjects are not able to recognize the surface they are stepping upon, they never confuse its nature, in the sense that they never select a solid surface when exposed to an aggregate one.

In addition, Tables 1, 2 and 3 show the degree to which participants judged the realism and quality of the experience. The degree of realism was calculated by looking only at that data from correct answers, i.e., when the surfaces were correctly recognized. This choice was performed since we were interested in understanding whether the simulation of specific surfaces recognized by the subjects was satisfactory.

As far as the quality judgement is concerned, the data was based on all the answers different from "I don't know".

The mean of realism is higher for condition 2 compared to condition 1 for each surface with the exception of beach sand (which is almost equal). The analysis by means of a t-test reveals that such differences are statistically significant for snow (p = 0.01055) and forest underbrush (p = 0.005595).
Analogously, the mean of realism is higher for condition 2 compared to condition 3 for each surface with the exception of beach sand (which is almost equal). In particular, the differences are statistically significant for beach sand (p = 0.002568) and snow (p = 0.001938).
Moreover, the mean of realism is higher for condition 1 compared to condition 3 for each surface with the exception of forest underbrush, which is minor. Such differences are statistically significant for beach sand (p = 0.001302), and for forest underbrush (p = 0.03438), which, as said, is greater for experiment 3.

As regards the mean of quality, it is higher for condition 2 compared to condition 1, with statistically significant differences for all the surfaces with the exception of dry leaves: beach sand (p = 0.009619), gravel (p = 0.02169), snow (p = 0.0006874) and forest underbrush (p = 0.02198).
The mean of quality is higher for condition 2 compared to condition 3 for each surface, and in particular the differences are statistically significant for beach sand (p = 0.006187), for snow (p = 9.596e-05).
Furthermore, the mean of quality is similar for condition 1 compared to condition 3, with the exception of forest underbrush for which it is higher in condition 3 compared to condition 1, with statistically significant differences (p = 0.03204).

	% Correct answers	% Wrong answers	% "I don't know"	Realism	Quality
Beach Sand	50.	46.67	3.33	5.2	4.7241
Gravel	83.33	6.67	10.	5.2	4.6296
Snow	73.33	26.67	0.	5.2955	5.1167
Forest Underbrush	40.	50.	10.	3.5	4.1923
Dry Leaves	16.67	63.33	20.	4.4	3.9167

Table 1. Results of condition 1: recognition of the surfaces without soundscapes.

	% Correct answers	%Wrong answers	% "I don't know"	Realism	Quality	% Correct soundscape
Beach Sand	86.67	10.	3.33	5.1481	5.5172	93.33
Gravel	86.67	13.33	0.	5.3077	5.4	86.67
Snow	80.	20.	0.	6.1667	6.0667	83.33
Forest Underbrush	73.33	26.67	0.	4.9091	5.0333	100.
Dry Leaves	30.	70.	0.	4.4444	4.5	96.67

Table 2. Results of condition 2: recognition of the surfaces with coherent soundscapes.

The comparison about the percentages of "I don't know" answers reveals that for each surface they are higher for condition 1 compared to condition 2, and for condition 3 compared to condition 2. In addition, they are higher for condition 3 compared to condition 1, for each surface with the exception of forest underbrush (which is minor).

As regards the percentages of correct answers about the soundscapes presented, they are higher for condition 2 compared to condition 3, and in particular the differences are statistically significant for the ski slope soundscape (p = 0.0003945).

Overall, subjects observed that soundscapes play an important role in recognition of the surfaces, precisely for their ability to create a context. Especially in terms of conflicting cues, as it was the case in condition 3, subjects were trying to identify the strongest cues, i.e. the element which had the strongest recognition factor. Sometimes the subjects found this task quite hard to complete, and this is why the percentage of "I don't know" answers is higher in condition 3 as opposed to condition 2.

When leaving a comment, several subjects observed that the recognition of snow was extremely realistic. This observation is also confirmed by the high degree of realism (mean = 5.3) and quality (mean = 5.1) with which such surface was rated.

On the other hand, for some subjects the concept of dry leaves was rather confusing, and this is also confirmed by the low recognition rate of such surface.

Overall, this experiment represents a strong indication

Material	Soundscape	% Correct answers	% Wrong answers	% No idea	Realism	Quality	% Correct soundscape
Beach Sand	Forest	38.46	57.7	3.84	4	5.16	88.46
Beach Sand	Ski slope	15.38	69.24	15.38	3.75	4.3182	38.46
Snow	Forest	50	42.31	7.69	5.2857	5.3542	96.15
Snow	Beach	50	46.16	3.84	4.7692	5	88.46
Forest Underbrush	Beach	38.46	53.85	7.69	4.8	4.9583	65.38
Forest Underbrush	Ski slope	30.76	65.4	3.84	4.3	4.6087	46.15

Table 3. Results of condition 3: recognition of the surfaces with incoherent soundscapes.

	BS	GL	SW	UB	DL	HG	DR	FS	WD	CW	MT	CC	PD	WT	CP	—
BS	15	2	6	2	2	2										1
GL		25			1								1			3
SW			22	1	1			6								
UB			4	12			1	10								3
DL		9		4	5			6								6

Legend:

WD	wood	CW	creaking wood	SW	snow	UB	underbrush
—	don't know	FS	Frozen snow	BS	beach sand	GL	Gravel
MT	metal	HG	High grass	DL	dry leaves	CC	concrete
DR	dirt	PD	puddles	WT	Water	CP	carpet

Table 4. Confusion matrix of condition 1.

	BS	GL	SW	UB	DL	HG	DR	FS	WD	CW	MT	CC	PD	WT	CP	—
BS	26	1	1	1												1
GL		26			3	1										
SW	2		24					4								
UB		1	2	22	1			4								
DL	6			9	9			6								

Legend:

WD	wood	CW	creaking wood	SW	snow	UB	underbrush
—	don't know	FS	Frozen snow	BS	beach sand	GL	Gravel
MT	metal	HG	High grass	DL	dry leaves	CC	concrete
DR	dirt	PD	puddles	WT	Water	CP	carpet

Table 5. Confusion matrix of condition 2.

	Soundscape	BS	GL	SW	UB	DL	HG	DR	FS	WD	CW	MT	CC	PD	WT	CP	—
BS	Forest	10		2	5	3	1	1	3								1
BS	Ski slope	4	1	5		1	5	3	1					1	1		4
SW	Forest		5	13	1	1			4								2
SW	Beach	2	6	13	2				2								1
UB	Beach		3	4	10	2			5								2
UB	Ski slope		1	1	8	3	1	2	8							1	1

Legend:

WD	wood	CW	creaking wood	SW	snow	UB	underbrush
—	don't know	FS	Frozen snow	BS	beach sand	GL	Gravel
MT	metal	HG	High grass	DL	dry leaves	CC	concrete
DR	dirt	PD	puddles	WT	Water	CP	carpet

Table 6. Confusion matrix of condition 3.

of the importance of context in the recognition of a virtual auditory place, where self sounds created by users' footsteps and soundscapes are combined. Further investigations are needed to enhance the realism of the simulated soundscape, in particular by having the auditory cues changing according to the motion of the subject in the space.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we describe an experiment whose goal is to understand the role of soundscapes in creating a sense of place and context when designing a virtual walking experience. In this particular experiment, the user was not able to interact with the soundscapes, which were made of mere soundtracks. The results described are an interesting starting point for further investigations on the role of environmental sounds to create a sense of place. While walking an acting in an environment, a person is exposed to her own self-sounds as well as the sounds of the place. This paper presents a preliminary investigation of the role of these different elements both taken in isolation and combined. Further investigations are needed to gain a better understanding of the cognitive factors involved when subjects are exposed to different sound events, especially when a situation of semantic incongruence is present.

We are also planning to design the soundscapes in a multichannel environment, where moving sound sources are present, and the location of the different sound events depends on the location of the subjects. We are also planning to enhance the simulations with visual and haptic feedback. This will allow us to investigate in depth the role of sound to create a sense of place in unimodal and multimodal environments.

5. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community' s Seventh Framework Programme under FET-Open grant agreement 222107 NIW - Natural Interactive Walking. [7]

6. REFERENCES

[1] M. Chion, C. Gorbman, and W. Murch, *Audio-vision: sound on screen*. Columbia Univ Pr, 1994.

[2] R. Schafer, *The tuning of the world*. Alfred A. Knopf, 1977.

[3] B. Blesser and L. Salter, *Spaces speak, are you listening?: experiencing aural architecture*. MIT Press, 2006.

[4] R. Storms and M. Zyda, "Interactions in perceived quality of auditory-visual displays," *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 6, pp. 557–580, 2000.

[5] R. Sanders Jr, *The effect of sound delivery methods on a users sense of presence in a virtual environment*. PhD thesis, NAVAL POSTGRADUATE SCHOOL, 2002.

[7] www.niwproject.eu

[6] P. Chueng and P. Marsden, "Designing Auditory Spaces to Support Sense of Place: The Role of Expectation," in *CSCW Workshop: The Role of Place in Shaping Virtual Community*, Citeseer, 2002.

[7] R. Nordahl, "Increasing the motion of users in photorealistic virtual environments by utilizing auditory rendering of the environment and ego-motion," *Proceedings of Presence*, pp. 57–62, 2006.

[8] L. Turchet, S. Serafin, S. Dimitrov, and R. Nordahl, "Physically based sound synthesis and control of footsteps sounds," in *Proceedings of Digital Audio Effects Conference*, 2010.

[9] R. Nordahl, L. Turchet, and S. Serafin, "Sound synthesis and evaluation of interactive footsteps for virtual reality applications," in *Proceedings of IEEE Virtual Reality*, 2010.

[10] S. Serafin, L. Turchet, and R. Nordahl, "Extraction of ground reaction forces for real-time synthesis of walking sounds," in *Proc. Audiomostly*, 2009.

AN AUDIOVISUAL WORKSPACE FOR PHYSICAL MODELS

Benjamin Schroeder, Marc Ainger, Richard Parent
The Ohio State University
benschroeder@acm.org, ainger.1@osu.edu, parent@cse.ohio-state.edu

ABSTRACT

We present an experimental environment for working with physically based sound models. We situate physical models in an interactive multi-modal space. Users may interact with the models through touch, using tangible controllers, or by setting up procedurally animated physical machines. The system responds with both real-time sound and graphics. A built-in strongly-timed scripting language allows for a different kind of exploration. The scripting language may be used to play the models with precise timing, to change their relation, and to create new behaviors. This environment gives direct, concrete ways for users to learn about how physical models work and begin to explore new musical ideas.

1. INTRODUCTION

Physically based sound synthesis is, for the user, both familiar and richly expressive. Physical models correspond to real-world objects, and variations in instrument design and playing style may be specified using real-world concepts like shape, forces, and material properties.

In this paper we describe an experimental environment for working with physical models. Interaction in our environment is direct and concrete; it corresponds to real-world experiences but goes beyond the strictly physical.

There are several ways the model can be affected. The models are situated in a virtual space and may be played using touch. Tangible controllers may be used to influence the model or the space they reside in. Small "machines" based on procedural animation and physics give another way to explore relations between space, time, and rhythm. A textual scripting language provides for more precise timing, deeper exploration, and extension of the system's behavior.

Our environment is multi-modal: models respond both aurally and visually in real time. We support touch input (as well as conventional mouse input) and input from different kinds of tangible controllers. We intend for the users' different senses - auditory, visual, and kinesthetic - to work together to create a fuller interaction experience.

This work is certainly inspired by other sound systems which use interactive touch, most notably the Reactable [1]. The Reactable situates sound objects in spatial relation and uses space to create rhythmic patterns. Its objects are based on signal processing and abstract notation; our research uses physically based models and a more concrete, physical notion of space.

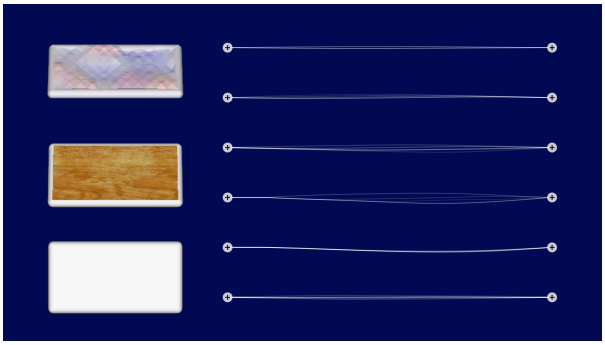


Figure 1. Several strings and plates in the direct-manipulation environment.

In what follows, we introduce the different kinds of interaction supported by our system: direct manipulation, tangible controllers, procedural animation, and textual scripting. We then conclude with a brief discussion of questions to be addressed in future research.

1.1 Video and audio examples

Video examples (with sound) demonstrating several features of our system are available on the web at *http://www.youtube.com/user/avworkspacesmc2010*.

2. SPATIAL, MULTI-MODAL INTERACTION

Physical models, like their real-world counterparts, are situated in time and space. They make sound when things interact with them in physical ways: plucking, bowing, striking, fretting. This suggests a graphical setting in which models may be arranged spatially and interacted with using direct manipulation.

Our environment includes strings and plates as sounding primitives. Figure 1 shows several strings and metal plates. A user can play the strings by "plucking" them with the mouse. Moving across several strings at once produces a strum. Tapping on a plate makes a clanging sound.

Since the strings and the plate are implemented using physical models, they respond realistically to differing input. For example, real strings and plates produce different sounds depending on where they are plucked or struck due to the excitation of different vibrational modes. This is true

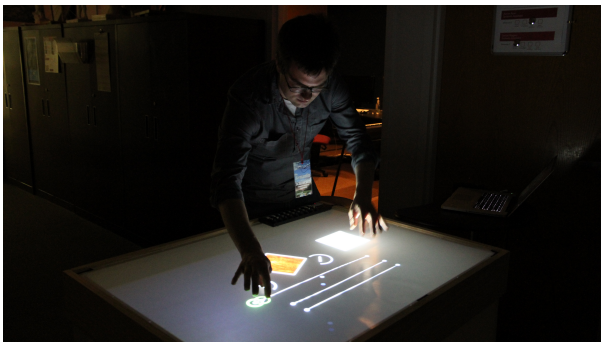


Figure 2. The system running on a diffuse-illumination multitouch table.

of our models as well.

Sound in a physical model comes from the simulation of an object's vibration. The models in our system respond visually as well as sonically, giving users a way to build intuition about how changes in shape produce changes in sound. (The visual response may be exaggerated in scale far beyond what is realistic.) Time in the system may be stretched into slow motion, allowing for close examination of how waves progress through an object. Most of the graphics in our system are shown in a 2D diagrammatic perspective, but plates may be tilted in 3D to provide a better view of their complex vibrations.

Models may also be changed through simple direct manipulation. Strings and plates may be stretched to change their size and hence the sound they produce. String tension may be set by manipulating a string's end as though using a tuning peg. All of these changes are made interactively; changes can even be made while an object is sounding.

A multitouch surface is a natural setting for this kind of interaction, and we have experimented with running our system on a diffuse-illumination multitouch table. (Our table is shown in Figure 2; a brief description of the technology is given in Appendix B.) Although touch is not strictly necessary for our system, the availability of multiple touches gives rise to additional kinds of interaction, such as fretting a string to produce different notes. Furthermore, the table setting allows multiple users to play at once.

2.1 Physical model implementation

Our string and plate primitives are implemented using finite differences [2]. In principle, any physical modeling technique could be used, as long as it accepts input in terms of forces and positions and provides output in these same terms.

Finite difference models work especially well in this regard; physical quantities are calculated for each point on a finite-difference grid at every time step, making it trivial to use a model's output for such things as real-time visualization. Input is in terms of these same physical quantities. By contrast, frequency-based techniques such as modal synthesis accept input in terms of forces, but require more computation in order to provide output in terms of physical quantities rather than fully synthesized waveforms.

We use a string model described by Chaigne and Askenfelt [3] and a plate model given by Bilbao [4]. Our string plucking model is a partial implementation of that described by Cuzzucoli and Lombardo [5]. The main equations for the string and plate are reproduced here for convenient reference and in order that we might mention the available parameters; for full details, please see the appropriate papers.

The string model is given by the following equation, which describes a string's basic motion as well as internal and radiative damping. The equation also models dispersion due to stiffness, as occurs, for example, in piano strings.

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2} - \epsilon c^2 L^2 \frac{\partial^4 y}{\partial x^2} - 2b_1 \frac{\partial y}{\partial t} + 2b_3 \frac{\partial^3 y}{\partial t^3} + f(x, x_0, t). \quad (1)$$

In this equation, c is the speed of sound on the string, which incorporates tension and the string's mass density; L is the string's length; b_1 and b_3 are damping constants. The coefficient ϵ describes the string's stiffness. The function f accounts for force interaction over time.

The plate model is similar but is given in two dimensions.

$$\frac{\partial^2 u}{\partial t^2} = -\kappa^2 \nabla^4 u + c^2 \nabla^2 u - 2\sigma \frac{\partial u}{\partial t} + b_3 \frac{\partial}{\partial t} \nabla^2 u + f(x, y, t). \quad (2)$$

Here, κ describes the plate's stiffness; c is again the speed of sound due to any applied tension. The coefficients σ and b_3 are damping constants [1]. The function f again represents force interaction.

3. TANGIBLE CONTROLLERS

We have discussed one way to influence the system of sounding objects: using touch and spatial motion. Another way is to use tangible physical controllers and sensors. These expand the expressivity of the system, giving us ways to map things like 3D motion, shape, pressure, or temperature into our environment. In addition, such controllers have a satisfying physical presence; even simple controls like sliders and buttons can seem more "real" when made out of actual plastic and steel rather than pixels.

Our environment can receive messages directly from MIDI controllers and Wii remotes. It can also receive Jitter network messages, allowing for the use of Max/MSP/Jitter as a sort of pre-processing frontend. Messages of this sort can easily be made to control the parameters of a model, such as its tension, length, or position, or to induce actions such as plucks.

More interestingly, controllers might be used to affect the physical environment in broader ways. Figure 3 shows a breath controller given a virtual presence. It may be moved and turned like any other object via direct manipulation. Blowing into the controller causes a "wind" of particles to enter the space. The particles pluck the strings and

[1] Bilbao gives this coefficient as b_1, but we have here used b_3 for consistency with the string model.

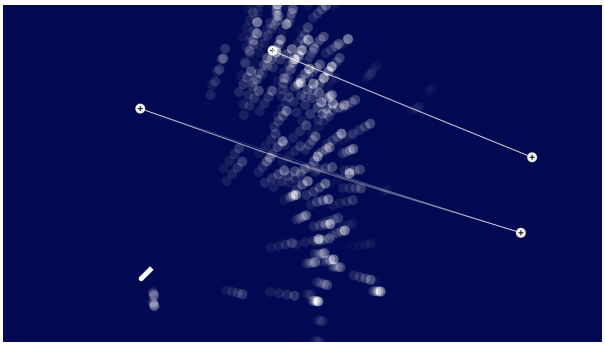


Figure 3. A "wind" of particles from a breath controller.

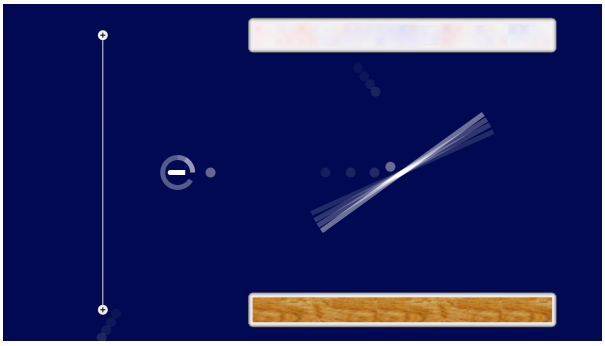


Figure 4. A basic procedural machine.

bounce off the plates. We might now consider using a camera system and digital compass to map a person's physical location and orientation into the system, letting them in a sense move through the simulated space, blowing wind as they go.

4. PROCEDURAL MACHINES

Procedural animation techniques introduce more variety into the graphical environment and give ways to explore the role of algorithms in sound generation. We have already seen one example of this: the particle system of the breath controller's "wind". As encapsulated algorithms, procedural animations also allow for actions to be repeated in more precise ways than direct interaction does, and for the actions to be edited over time. This supports an iterative style of design.

Figure 4 shows a basic procedural machine. In this system, there is an emitter on the left that sends particles into the environment at a regular rate, like a metronome. The slab to its right rotates back and forth, bouncing the particles either to the top plate, made of metal, or to the bottom plate, made of wood. Occasionally a particle ricochets off the slab head-on and plucks the string behind the emitter. By controlling the rate and angle of particle emission, the user can make different rhythmic patterns and learn about relationships between organized space, velocity, and time.

Other kinds of emitters are possible. For example, we have built objects that split incoming particles into two. Gravity (in any direction and strength) adds another twist

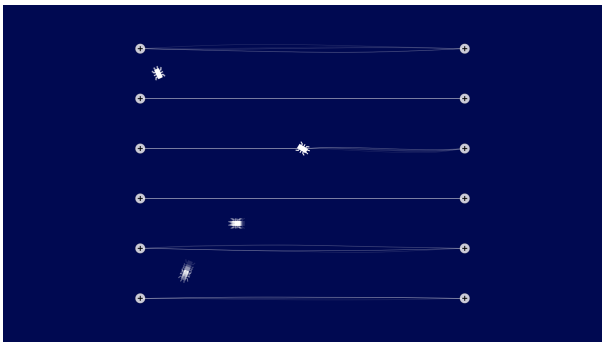


Figure 5. Procedural crawlers which fret strings.

to the possibilities of particle motion.

Physics is not the only way to interact algorithmically with the system. In Figure 5, small robots crawl around the space, moving to random locations on a grid. When a crawler moves to a string, it latches on, fretting the string for a time, before moving off in a random direction again.

5. INTERACTIVE PROGRAMMING

So far, we have described several high-level ways to interact with sound models. Our environment also includes a textual language which provides for low-level access to the models. This has two purposes. First, textual scripting is a flexible way to set up direct-manipulation environments like those described above and to describe new procedural elements. Scripts, like procedural animations, support iterative design by allowing for the reuse and careful editing of algorithms.

Second, while direct manipulation is approachable and concrete, text is a good choice for describing more sophisticated interactions. In particular, it is more apparent how to use parameterization and abstraction in this context, and timing may be made more precise.

The textual language is interactive and runs in conjunction with the direct manipulation environment. This allows a user to go back and forth between the two as desired.

A complete reference is beyond the scope of this paper. However, the examples that follow are intended to give an idea of how the language works. We introduce the examples with a short discussion of the textual language's general structure and philosophy.

5.1 Language structure and philosophy

The textual language is a prototype-based object-oriented language; it is similar in many ways to Self [6] or Smalltalk. Our concern in this environment is mainly for ad-hoc and on-the-fly programming, rather than the construction of larger software systems. We have therefore emphasized programming facilities that support experimentation, rather than ones that support longer-term efforts. Although the underlying language is capable of representing complex abstractions, we have mainly concentrated on its use in an interactive context.

Figure 6. Scripting takes place through an interactive workspace.

Most interaction with the language happens through a text window called a "Workspace" (Figure 6). This is similar to the interactive command line or read-eval-print loop provided by some environments, but it retains more context from step to step. Code in a Workspace can be executed line-by-line; results may be printed; old code can be revisited. In keeping with the philosophy of supporting experimentation, variables set at the top level of our Workspaces are defined automatically, allowing users to define names as they go.

The textual environment runs concurrently with the direct-manipulation environment and with audio output. In this way, results from code execution can be seen immediately, and changes in the output can be investigated more closely using code. Users can move freely between the two levels as needed; in a multi-user setting, one person might even be playing an instrument using touch while the other modifies it using code.

Our language follows ChucK [7] in being *strongly timed*. The simulation time for a given thread proceeds (and audio samples are calculated) only when the programmer explicitly asks for it to do so. Code between such statements is considered, from the perspective of the simulation, to execute instantaneously. In this way, the programmer is given full control over timing and coordination.

In our environment, it is possible that no user-level code is running at some particular time, but instead that all interaction is taking place through the direct-manipulation environment. In that case, simulation time proceeds along with real time.

5.2 A simple example

Imagine that you wanted to try playing some notes on plucked strings. You might start by making a string and tuning it.

```
stringD: stringModel make.
stringD frequency: 146.80.
```

This puts a string, tuned to D, on the screen, and assigns it the variable name "stringD". You can pluck the string by hand or using code.

```
stringD pluckAtFraction: 0.7.
```

We can add a second string, positioning it a little below the first. Plucking the strings at timed intervals while changing their notes plays a familiar tune. (Quotes are used below to add comments.)

```
stringA: stringModel make.
stringA frequency: 110.
stringA center: 0 @ -0.01.

stringD fretAtIndex: 2. "E"
stringD pluckAtFraction: 0.7.
simTime advance: 0.25 seconds.

stringD fretAtIndex: 0. "D"
stringD pluckAtFraction: 0.7.
simTime advance: 0.25 seconds.

stringA fretAtIndex: 3. "C"
stringA pluckAtFraction: 0.7.
simTime advance: 0.25 seconds.

stringD pluckAtFraction: 0.7.
simTime advance: 0.25 seconds.

stringD fretAtIndex: 2. "E"
3 timesRepeat:
  [stringD pluckAtFraction: 0.7.
  simTime advance: 0.25 seconds].
```

5.3 Making physical changes

One of the strengths of many physical models is their ability to represent different kinds of material. Strings are created by default as nylon strings. We can change the strings above to be steel strings by assigning new material properties.

```
{stringA. stringD} do:
  [:each |
  each
    changeMaterialDensity: 7800.0
    youngsModulus: 200e9].
```

For convenience, this keeps the strings at the same frequency as before by adjusting their tension to match the new material. The strings will sound slightly different due to their new material properties and tension. Other properties of the models, such as damping coefficients, may be changed as well.

5.4 Adding new behaviors

The lengths of the strings in the direct manipulation environment can be changed; this also changes their frequency. The tune in the code above is written in terms of the built-in frets, which are proportional to the length of the string (they are set at half-steps, like guitar frets). Therefore, changing the length of the string would transpose the tune up and down; this could be done while the tune was playing to act as a sort of simple (if not strictly realistic) tremolo arm (or "whammy bar").

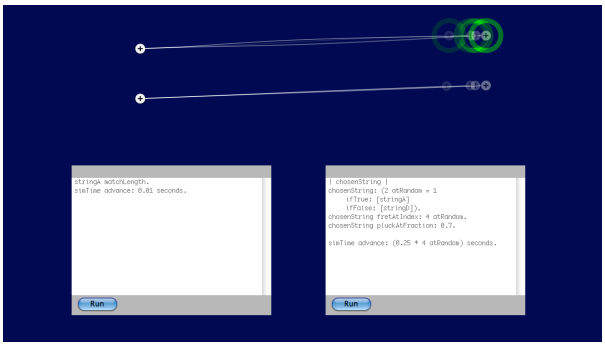


Figure 7. Facilities for running code in continuous loops.

However, the tune is written on two strings. It would be difficult to change these in exactly the same way simply by dragging. We can keep the second string up to date with the first using the following code.

```
stringA learn:
  (|
  partner <- nil.
  matchLength = (| delta |
    delta: partner endPoint
      - partner startPoint.
    endPoint: startPoint + delta).
  |).
stringA partner: stringD.

[[true] whileTrue:
  [stringA matchLength.
  simTime advance: 0.01 seconds]] fork.
```

This code first teaches the A string a new behavior, how to match length with a partner. It then creates a new independent thread which asks the string to match length every hundredth of a second.

Note that this acts like a constraint, but it is only an approximation of one. In particular, it is not updated continuously, and the custom matching code could be expensive to run frequently. A facility for constraints is a topic for future research.

Forking new continuously-running threads is useful for many things. When working with threads, it is important to be able to stop and start them at will, while retaining their code. (In the example above, the thread never stops.) The graphical environment therefore has a facility for running continuously-looping code, as shown in Figure 7.

One could envision a way to express the length-matching behavior via direct manipulation, such as attaching all of the strings to a rigid bar and then moving the bar: a kind of virtual tremolo arm to match the behavior. The scripting environment lets us prototype new ideas before adding them at higher levels.

5.5 New procedural objects

Code may also be used to create new kinds of objects for the procedural animation environment. Figure 8 shows a simple but complete example of an emitter that sends

```
firework: proceduralElement make.
firework learn:
  (|
  emit = (| newBall. theta |
    newBall: ball make.
    newBall radius: 0.01.
    newBall dragFactor: 0.01.
    newBall beFading.
    newBall position: position.

    theta: ((24 atRandom) * 15.0)
      asRadians.
    newBall impulse:
      (theta cos @ theta sin)
      * (0.1 @ 0.1)).

  burst = (||
    40 timesRepeat:
      [| waitTime |
      emit.

      waitTime: (5 + 5 atRandom)
        milliseconds.
      simTime advance: waitTime]).
  |).
```

Figure 8. A "fireworks" animation object.

a burst of particles into the space at random angles and speeds. It may be activated by executing `fireworks burst`.

6. CONCLUSION AND FUTURE WORK

We have described our interactive environment for experimenting with physical models in space and in code. The environment supports several different kinds of interaction with the models, from direct playing to the use of procedural animation and physics and exploration through code.

Our goal with this environment has been to provide for easy, high-level experimentation with physical models. However, the range of expressiveness of physical models is deep, and we have only scratched the surface.

For example, we might want facilities for creating new models of a player's interaction with a string: different kinds of plucks, bowing, fretting models, playing string harmonics, rasgueado, and more. We might consider physical models that change over time in unrealizable but physically plausible ways, such as a web of strings that shifts around, splitting and reconnecting over time. Models might be connected to one another and resonate. Input from more sophisticated tangible controllers or even other instruments might be used to drive models. In addition, several musically important concepts addressed in existing languages go unaddressed here.

In our future research, we hope to address deeper questions such as these. We expect that new abstractions, methods of interaction, and language facilities will help make the full power of physical models truly accessible.

7. REFERENCES

[1] M. Kaltenbrunner, S. Jorda, G. Geiger, and M. Alonso, "The reactable*: A collaborative musical instrument," *Enabling Technologies, IEEE International Workshops on*, pp. 406–411, 2006.

[2] S. Bilbao, *Numerical Sound Synthesis*. Chichester: John Wiley & Sons, Ltd., 2009.

[3] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I.," *Journal of the Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[4] S. Bilbao, "A finite difference plate model," in *Proc. of Intl Comp. Music Conf. (ICMC 2005)*, 2005.

[5] G. Cuzzucoli and V. Lombardo, "A physical model of the classical guitar, including the player's touch," *Comput. Music J.*, vol. 23, no. 2, pp. 52–69, 1999.

[6] D. Ungar and R. B. Smith, "Self: The power of simplicity," *SIGPLAN Not.*, vol. 22, no. 12, pp. 227–242, 1987.

[7] G. Wang and P. R. Cook, "ChucK: a concurrent, on-the-fly audio programming language," in *Proc. ICMC*, pp. 219–226, 2003.

[8] E. Catto, "Box2D," 2010. http://www.box2d.org. Accessed April 22, 2010.

[9] NUI Group Community, "Community core vision," 2010. http://ccv.nuigroup.com/. Accessed April 27, 2010.

[10] Hiroaki *et al.*, "DarwiinRemote," 2008. http://sourceforge.net/projects/darwiin-remote/. Accessed April 22, 2010.

[11] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza, "Tuio: A protocol for table-top tangible user interfaces," in *6th International Gesture Workshop*, 2005.

[12] NUI Group Authors, "Multi-touch technologies," 2009. http://nuicode.com/projects/wiki-book. Accessed April 27, 2010.

[13] G. P. Scavone and P. R. Cook, "RTMidi, RTAudio, and a Synthesis Toolkit (STK) update," in *In Proceedings of the International Computer Music Conference*, 2005.

A. IMPLEMENTATION NOTES

Our system runs in real time on commodity Macintosh hardware. We make use of several open-source libraries: RtMidi and RtAudio [13], DarwiinRemote [10], and the Box2D physics engine [8]. Our multi-touch implementation uses the CCV tracking system [9] and the TUIO multitouch protocol [11].

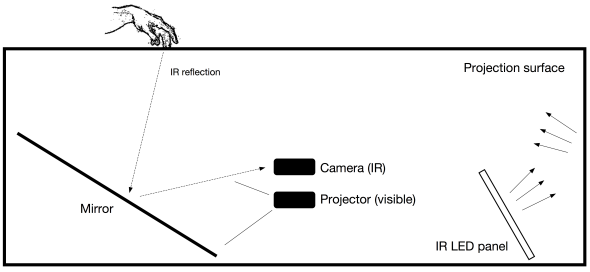


Figure 9. A diffuse-illumination multitouch table.

B. DIFFUSE-ILLUMINATION MULTITOUCH

One setting for our system has been a diffuse-illumination multitouch table. This appendix contains a brief discussion of how such tables may be implemented. A more complete description of this and other multitouch implementation techniques may be found in an excellent online book by the NUI Group [12].

Multitouch tables allow for direct interaction with displayed images. A major question in the implementation of such tables is therefore how to detect touches. Diffuse-illumination tables detect touches via a camera system; to distinguish between actual touches and changes in the display, they rely on the difference between visible and infrared light.

As shown in Figure 9, both a projector and a camera are mounted below the table surface. A mirror may be used (as shown here) to allow for larger projections without requiring the table to be very deep.

The projector displays a video image on a projection surface at the top of the table. This image is made up only of visible light; commodity projectors typically do not emit any infrared light. A number of infrared LED panels also illuminate the projection surface. Since these only emit light in the infrared spectrum, they are invisible to users of the table. The illumination is meant to be spread evenly across the surface. To some extent, the light is diffused by the surface itself, which is semi-opaque; to further avoid "hot spots" of illumination, light is often bounced off of a wall or other surface before reaching the top of the table.

The camera is equipped with an infrared-passing filter; it only "sees" infrared light and does not react to changes in the visible part of the spectrum. If nothing is touching the table, then, the camera sees a static image, even if the video image displayed by the projector is changing.

If a user touches the table, light reflects more strongly from the location of the touch. Techniques from computer vision are used to isolate the area of the touch from the surrounding static background. Multiple touches are seen as separate areas in the resulting image. These touches are tracked over time; their locations and shapes are passed to higher-level software such as our system for further processing.

COMPARISON OF NON-STATIONARY SINUSOID ESTIMATION METHODS USING REASSIGNMENT AND DERIVATIVES

Sašo Muševič
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
saso.musevic@upf.edu

Jordi Bonada
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
jordi.bonada@upf.edu

ABSTRACT

In this paper, three state of the art non-stationary sinusoidal analysis methods based on Fourier transform (FT) are compared - the derivative method, reassignment and generalized reassignment. [1] The derivative method and reassignment were designed to analyze linear log-AM/linear FM sinusoids. Generalized reassignment can analyze sinusoids containing arbitrary order modulations, however the discussion will be limited to linear log-AM/linear FM in order to compare it objectively to reassignment and the derivative method. In this paper, the equivalence of reassignment and the derivative method is shown to hold for arbitray order modulation estimation and theoretical comparison with generalized reassignment is presented. The results of tests conducted on two different frequency ranges, full range (frequencies up to Nyquist) and reduced range (frequencies up to 3/4 Nyquist) frequency range, are compared to the Cramer-Rao bounds (CRBs).

1. INTRODUCTION

Sinusoidal modeling of sound signals is used in many audio analysis/synthesis applications [1],[2],[3]. Several analysis methods for estimating sinusoidal model parameters based on Short Time Fourier Transform (STFT) assume that the underlying sinusoid is quasi-stationary inside a selected time frame [4],[5],[6]. Since real world sounds often violate this assumption, the analysis methods able to detect first order polynomial modulations have received much attention [7],[8],[9],[10],[11]. It is straightforward to see [12] that the reassignment method could be theoretically generalized to detect higher order polynomial modulations for both log-AM and FM. As it was shown in [8] and [13] that the reassignment and the derivative methods are theoretically identical, then the same must hold for the derivative method. However, expressions for estimating higher order modulations get very complex in both cases and its derivations are not straightforward. The theoretical equality of both methods can be exploited to construct the generalized reassignment method [14], which estimates parameters of signal modulations up to an arbitrary degree, although some restrictions concerning window function apply. In [15] the generalized reassignment is shown to work for any linear transform, which includes the STFT, the wavelet transform or even a combination of them.

The derivative method, reassignment and generalized reassignment are all based on the same theoretical background, yet perform quite different in practice. Therefore, a detailed comparison of its internal mechanics is performed in the present document. In section 2, the common theoretical background including general equations for parameter estimations is derived. The differences between reassignment and the derivative method are described and method-specific parameter estimate expressions for the two are derived from the general ones, providing a mathematically identical proof already given in [8] and [13], extended to an arbitrary modulation degree. In section 3 the theoretical differences between original and generalized reassignment are outlined. Section 4 describes the test environment and summarizes accuracies collected in tests of all three estimators and compares them to CRBs. Conclusions and future work suggestions are given in section 5.

2. REASSIGNMENT AND THE DERIVATIVE METHOD

This section demonstrates that reassignment and the derivative method are two versions of the same algorithm. This fact was already pointed out in [8] and [13]. Present derivations prove that the derivative method is essentially a reassignment method with a slightly modified STFT definition or vice versa. In the following expression, the STFT definitions for the derivative method (D) and reassignment (R) are given respectively:

$$STFT\big(s(t), w(t); t, \omega\big) = {}^{\mathrm{D}}S_w(t,\omega) = \int_{-\infty}^{\infty} s(\tau + t) w(\tau) e^{-j\omega\tau} d\tau \qquad (1)$$

[1] The latter method was not explicitly named in [14] where it was first presented, therefore the authors decided on the name *generalized reassignment*, because the method exhibits similarity with the original reassignment and can successfully analyze a *generalized sinusoid* (a sinusoid with arbitrary order polynomial log-AM and FM function), a term adopted from [15]. The method exhibits close relation to the derivative method as well, however the name *generalized derivative method* could cause ambiguity with the method described in [8].

$$STFT\big(s(t),w(t);t,\omega\big) = {}^{\text{R}}S_w(t,\omega) = \int_{t-\infty}^{t+\infty} s(\bar{\tau})w(\bar{\tau}-t)e^{-j\omega(\bar{\tau}-t)}d\bar{\tau}, \quad (2)$$

where $s(t), w(t)$ are signal and window functions respectively. It is obvious, that the second definition can be derived from the first by substituting the variables: $\tau = \bar{\tau} - t$. Essentially ${}^{\text{R}}S_w = {}^{\text{D}}S_w$; however, its the time/frequency derivatives yield different expressions (see Appendix A), yet a common notation S_w will be used for both ${}^{\text{R}}S_w$ and ${}^{\text{D}}S_w$ from this point on.

S_w is a function of time and frequency which can be written in polar form as:

$$S_w(t,\omega) = \exp\big(a(t,\omega) + j\phi(t,\omega)\big). \quad (3)$$

From the above representation, amplitude and phase functions of time and frequency can be written as:

$$a(t,\omega) = \Re\Big(\log\big(S_w(t,\omega)\big)\Big) \quad (4)$$

$$\phi(t,\omega) = \Im\Big(\log\big(S_w(t,\omega)\big)\Big). \quad (5)$$

Reassigned frequency and time equations can be expressed for this general case as:

$$\hat{\omega}(t,\omega) = \frac{\partial}{\partial t}\phi(t,\omega) = \Im\left(\frac{\frac{\partial S_w}{\partial t}}{S_w}\right) \quad (6)$$

$$\hat{t}(t,\omega) = t - \frac{\partial}{\partial \omega}\phi(t,\omega) = t - \Im\left(\frac{\frac{\partial S_w}{\partial \omega}}{S_w}\right). \quad (7)$$

Linear log-AM/linear FM are commonly defined in the following form:

$$s(t) = \exp\big(\lambda_0 + \mu_0 t + j(\phi_0 + \omega_0 t + \frac{\psi_0}{2}t^2)\big). \quad (8)$$

As pointed out in [8] and [10], general log-AM and FM expressions can be written as:

$$\hat{\mu}(t,\omega) = \frac{\partial}{\partial t}a(t,\omega) = \Re\left(\frac{\frac{\partial S_w}{\partial t}}{S_w}\right) \quad (9)$$

$$\hat{\psi}(t,\omega) = \frac{\partial\hat{\omega}}{\partial\hat{t}} = \frac{\partial\hat{\omega}}{\partial t}\Big/\frac{\partial\hat{t}}{\partial t} \quad (10)$$

$$\frac{\partial\hat{\omega}}{\partial t} = \Im\left(\frac{\frac{\partial^2 S_w}{\partial t^2}S_w - \left(\frac{\partial S_w}{\partial t}\right)^2}{\left(S_w\right)^2}\right) \quad (11)$$

$$\frac{\partial\hat{t}}{\partial t} = 1 - \Im\left(\frac{\frac{\partial^2 S_w}{\partial\omega\partial t}S_w - \frac{\partial S_w}{\partial\omega}\frac{\partial S_w}{\partial t}}{\left(S_w\right)^2}\right). \quad (12)$$

The above equations provide estimate espressions independent of the method used and thus hold for both reassignment and the derivative method. Once linear log-AM, frequency and linear FM parameters are estimated, the following expressions can be used to obtain accurate estimates for the two static parameters [7],[8]:

$$\Gamma_w(\omega,\mu,\psi) = \int_{-\infty}^{\infty} w(t)\exp\left(\mu t + j\left(\omega t + \frac{\psi}{2}t^2\right)\right)dt \quad (13)$$

$$\hat{a}_0 = \left|\frac{S_w}{\Gamma_w(\omega_\Delta,\hat{\mu}_0,\hat{\psi}_0)}\right| \quad (14)$$

$$\hat{\phi}_0 = \angle\left(\frac{S_w}{\Gamma_w(\omega_\Delta,\hat{\mu}_0,\hat{\psi}_0)}\right). \quad (15)$$

In order to obtain the above expressions for the two methods, partial frequency and time derivatives of S_w should be computed for reassignment and the derivative method. In the following formulas, $S_{w'}$ and S_{tw} represent the STFT of a signal, but the window derivative and time-ramped window functions are used instead of the original ones respectively, while S'_w represents the STFT of the time derivative of a signal. For reassignment, the following expressions with some restrictions (see Appendix A) apply:

$$\frac{\partial}{\partial t}S_w = -S_{w'} + j\omega S_w \quad (16)$$

$$\frac{\partial}{\partial \omega}S_w = -jS_{tw} \quad (17)$$

$$\frac{\partial^2}{\partial\omega\partial t}S_w = jS_{tw'} + jS_w + \omega S_{tw} \quad (18)$$

$$\frac{\partial^2}{\partial t^2}S_w = S_{w''} - 2j\omega S_{w'} - \omega^2 S_w. \quad (19)$$

Detailed derivations of 16 and 17 can be found in Appendix A (see equations 39, 40 and 43). Equation 18 is derived in detail in Appendix A (see equation 41), however the generalized rule (see equation 42) for time derivatives can be used as well. Equation 19 can be derived by using equation 42 twice. For the derivative method, slightly simpler expressions hold:

$$\frac{\partial}{\partial t}S_w = S'_w \quad (20)$$

$$\frac{\partial}{\partial \omega}S_w = -jS_{tw} \quad (21)$$

$$\frac{\partial^2}{\partial\omega\partial t}S_w = -jS'_{tw} \quad (22)$$

$$\frac{\partial^2}{\partial t^2}S_w = S''_w. \quad (23)$$

Substituting reassignment STFT expressions 16-19 into general equations for parameter estimations 6-12 yields:

$${}^{\text{R}}\hat{\omega}(t,\omega) = \omega - \Im\left(\frac{S_{w'}}{S_w}\right) \quad (24)$$

$${}^{\text{R}}\hat{\mu}(t,\omega) = -\Re\left(\frac{S_{w'}}{S_w}\right) \quad (25)$$

$${}^{\text{R}}\hat{\psi}(t,\omega) = \frac{\Im\left(\frac{S_w S_{w''} - (S_{w'})^2}{(S_w)^2}\right)}{\Re\left(\frac{S_{tw'}S_w - S_{tw}S_w}{(S_w)^2}\right)}, \quad (26)$$

which are well known reassignment expressions for estimating parameters of log-AM/FM sinusoids. Analogously,

substituting derivative method STFT expressions 20-23 into same equations results in:

$$^{\text{D}}\hat{\omega}(t,\omega) = \Im\left(\frac{S_w'}{S_w}\right) \tag{27}$$

$$^{\text{D}}\hat{\mu}(t,\omega) = \Re\left(\frac{S_w'}{S_w}\right) \tag{28}$$

$$^{\text{D}}\hat{\psi}(t,\omega) = \frac{\Im\left(\frac{S_w''}{S_w}\right) - 2\,^{\text{D}}\hat{\mu}(t,\omega)\,^{\text{D}}\hat{\omega}(t,\omega)}{1 + \Re\left(\frac{S_{tw}'S_w - S_{tw}S_w'}{(S_w)^2}\right)}, \tag{29}$$

which are the derivative method expressions as given in [8] and [13]. Since expressions 27 and 28 are straightforward, only detailed derivations of equation 29 can be found in Appendix B (see equations 50, 51).

This section has clearly demonstrated that reassignment and the derivative method are in fact analogous methods, derived from the same general linear log-AM/linear FM equations. The only difference is the definition of STFT, which results in quite different expressions for parameter estimates. Mathematically identical proof was already given in [8] and [13], however it was given for each parameter of linear log-AM/linear FM sinusoids separately and thus does not prove the equivalence of the two methods for arbitrarily modulated sinusoids. In order to prove equivalence of the methods in such a general case, arbitrary order time derivatives of general linear FM parameter expressions (equation 10) should be considered: $\frac{\partial^n\hat{\omega}}{\partial\hat{t}^n} = \frac{\partial^n\hat{\omega}}{\partial t^n}/\frac{\partial^n\hat{t}}{\partial t^n}$. Such expressions would contain STFTs of the form $\frac{\partial^{k+l}S_w}{\partial t^k\partial\omega^l}$. By using the rules 42, 43, 44, 45 it is possible to transform the general expressions into reassignment ones, containing STFTs of the form $S_{w^{(k)}t^l}$ and analogously into the derivative method ones, containing STFTs of the form $S^{(k)}_{t^l}$. It is straightforward that reassignment and corresponding derivative method expressions are identical for all modulation degrees. The same procedure can be performed for log-AM, concluding the proof of equality of the two methods for an arbitrary modulated sinusoid.

The derivative method requires computation of signal time-derivatives, as opposed to reassignment, which requires computation of the window time-derivatives. In practice, it is impossible to avoid errors computing time derivative of the signal in time domain. For that purpose, a derivation filter is used, however unacceptable errors occur at high frequencies [8]. Further, using such filter increases the frame length requirements of STFT and raises computational complexity. When performing STFT, analytical expression for window function is known in most cases, therefore exact analytical expression for its time derivatives can generally be computed before performing STFT, which does not add any computational complexity. It can be concluded that lower computational complexity and higher accuracy is expected from the reassignment estimates compared to the derivative method ones. However, tests have shown that in the reduced frequency range (up to 3/4 Nyquist), methods perform comparably [8].

3. REASSIGNMENT AND GENERALIZED REASSIGNMENT

Generalized reassignment is the latest method based on the same backround as reassignment and the derivative method. The method is essentially based on the derivative method, as it uses signal derivatives for estimating the parameters. However, integration *per-partes* is used to transform expressions containing signal derivatives to expressions containing ramped window derivatives [14]. Final expressions resemble much more those of reassignment than those of the derivative method. Although it is designed to estimate arbitrary order log-AM/FM modulations, the discussion will be restricted to linear log-AM/FM signals. The following equations apply in this context (from [14]):

$$(^{\text{GR}}\hat{\mu}(t,\omega) + j\,^{\text{GR}}\hat{\omega}(t,\omega))S_w + j\,^{\text{GR}}\hat{\psi}(t,\omega)S_{tw} = -S_{w'} + j\omega S_w \tag{30}$$

$$S_{w''} - j2\omega S_{w'} - S_w\omega^2 = (^{\text{GR}}\hat{\mu}(t,\omega) + j\,^{\text{GR}}\hat{\omega}(t,\omega))(-S_{w'} + j\omega S_w) - {}^{\text{GR}}\hat{\psi}(t,\omega)S_{tw}. \tag{31}$$

From above equations, the following estimates can be expressed [14]:

$$^{\text{GR}}\hat{\psi}(t,\omega) = \frac{\Im\left(\frac{S_wS_{w''} - (S_{w'})^2}{(S_w)^2}\right)}{\Re\left(\frac{S_{w'}S_{tw} - S_{tw}S_w}{(S_w)^2}\right)} \tag{32}$$

$$^{\text{GR}}\hat{\mu}(t,\omega) + j\,^{\text{GR}}\hat{\omega}(t,\omega) = j\omega - \frac{S_{w'}}{S_w} - j\,^{\text{GR}}\hat{\psi}(t,\omega)\frac{S_{tw}}{S_w} \Rightarrow \tag{33}$$

$$^{\text{GR}}\hat{\omega}(t,\omega) = \overbrace{\omega - \Im\left(\frac{S_{w'}}{S_w}\right)}^{^{\text{R}}\hat{\omega}} - {}^{\text{GR}}\hat{\psi}(t,\omega)\overbrace{\Re\left(\frac{S_{tw}}{S_w}\right)}^{t_\Delta = \hat{t} - t} \tag{34}$$

$$^{\text{GR}}\hat{\mu}(t,\omega) = \overbrace{-\Re\left(\frac{S_{w'}}{S_w}\right)}^{^{\text{R}}\hat{\mu}} - {}^{\text{GR}}\hat{\psi}(t,\omega)\Im\left(\frac{S_{tw}}{S_w}\right). \tag{35}$$

The FM estimate expression is identical to that of original reassignment. The frequency and log-AM estimates on the other hand, contain additional terms compared to those of original reassignment. In the case of the frequency estimate, the additional term is ψt_Δ. The original frequency reassignment gives a frequency estimate at a reassigned time $\hat{t}$, so the time shift $t_\Delta = \hat{t} - t = \Re\left(\frac{S_{tw}}{S_w}\right)$ can be used to correct the frequency estimate (e.g.: *move* frequency estimate back to desired time), once FM is estimated. In the case of a log-AM estimate, no such time correction is present, as the log-AM is modeled to be constant. However, another term $\hat{\psi}\Im\left(\frac{S_{tw}}{S_w}\right)$ with no straightforward interpretation is present. It can be thought of as a correction of an error that presence of FM causes on AM estimation.

It will be shown in tests that this additional terms improve frequency and AM estimate significantly in high signal-to-noise (SNR) ratios.

4. TESTS AND RESULTS

For easier comparison, the test parameter set was choosen identical to the one in [8], with exception of a frequency range. The test frequencies (100 in total) were linearly distributed over two different frequency ranges: reduced frequency range, 20Hz-16538Hz (3/4 of Nyquist) and full frequency range, 20Hz-22050Hz (Nyquist), the results for each range are plotted separately. All other test parameters were linearly distributed in the following ranges: 7 different phase values in range $[-\pi, \pi]$, 5 log-AM values values in $[-100, 100]$ range and 5 different FM values in $[-10000, 10000]$. Hanning window of length 511 samples and sampling frequency of 44100Hz was used. The error variance was calculated using all parameter combinations. In the case of the derivative method, the derivation filter as described in [8] of length 1023 was used and the frame length was extended to 1533 samples ($511+2\frac{1023-1}{2}$). After convolution with the filter, only the middle 511 samples out of 1533 were kept to avoid edge effect.

In order to perform parameter estimations, all the algorithms require an initial frequency estimate, which is a consequence of the fact that STFT is a function of frequency and time. The initial frequency estimate is commonly acquired by taking the bin frequency of the magnitude spectrum peak. Once a frequency estimate is made (using the bin frequency of the peak), this estimate itself can be used to estimate other parameters, even to *re-estimate* the frequency itself. Such a procedure can be performed many times, thus iteratively obtaining presumably more accurate estimations in each iteration. In [8], the derivative method was tested in the following setting: the bin frequency of the spectrum peak is used to obtain a frequency estimate, which is in turn used to estimate log-AM/FM and finally static log-amplitude/phase, but the frequency itself is not reestimated. On the other hand, reassignment was tested by using the bin frequency of spectrum peak for all estimates. In the presented test results, all algorithms use a frequency estimate (not the inital bin frequency of the peak) to obtain all subsequent higher order, as well as static parameter estimates. For that, it is reasonable to expect that reassignment will achieve better results as reported in [8]. Further, the FM estimates of the derivative method as defined in [8] did not take into account the group delay, which was pointed out in [13], thus the improvement of the FM estimate of the derivative method is also expected.

The estimation errors of the derivative method are identical for log-amplitude, log-AM, phase and frequency to those presented in [8], tested in 20Hz-16538Hz (3/4 Nyquist) frequency range. The FM estimate improved as expected. Reassignment performs significantly better than reported in [8], which renders the accuracy difference between the two in the reduced frequency range negligible. Generalized reassignment performs superior for all parameter estimations except FM in this frequency range, where all three methods perform roughly the same. Using the full frequency range, estimate errors of the derivative method rise significantly, while the original reassignment performs even better. Generalized reassignment again performs significantly better in all cases and seems to be unaffected by frequency range changes. This suggests, that generalized reassignment achieves identical accuracy for all frequencies. It is important to note, that a significant accuracy difference between reassignment and generalized reassignment occurs in higher SNR region. For each parameter estimation, it is possible to define a SNR value, above which the accuracies of reassignment and generalized reassignment differ significantly. Below such SNR value however, the two methods perform equally.

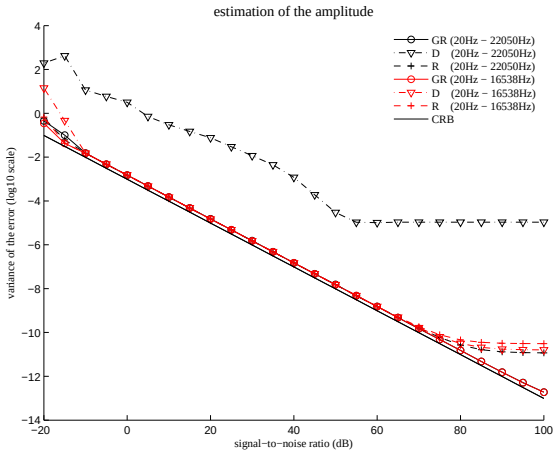
All three algorithms were implemented in Octave programming language and the tests were conducted with Octave 2.9.9 (x86-64 bit Debian distribution) on a Sun Grid Engine 6.2 (SGE) cluster.

5. CONCLUSION

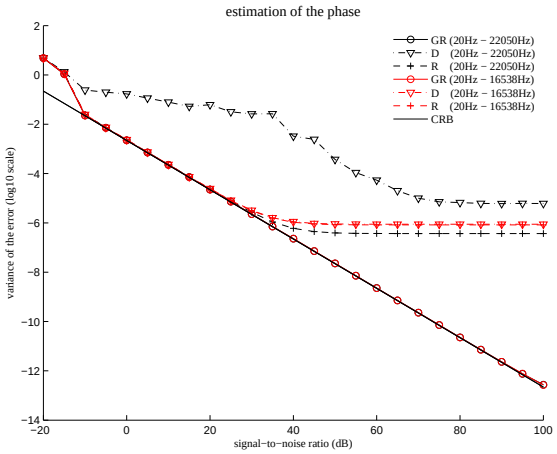
This document analyzes and compares three state of the art methods for non-stationary analysis and compares them to CRBs. It has been shown that generalized reassignment is the most appropriate method for analyzing complex linear log-AM/FM signals. Generalized reassignment is able to detect an arbitrary log-AM/FM degree of a sinusoid and achieves superior accuracy in the linear log-AM/linear FM case and should therefore be the topic of further research concerning the analysis of real world signals: both multicomponent and real. In this case, the degree of modulation of each sinusoid under study is unknown and determining its exact degree is crucial, as errors rise significantly when the modeled modulation degree is set either too high or too low [13]. Furthermore, the usual window functions like Hanning cannot be used when analyzing higher order modulations, as its second time derivative does not reach 0 at the start and end of the frame (required by the algorithm). Some higher order window functions were proposed in [13], however its exact accuracy currently remains unknown; therefore a more detailed study of window functions satisfying the restrictions of the algorithm should be researched. It is reasonable to expect, that such window functions would exhibit less desirable time-frequency trade offs. To avoid accuracy deterioration, limiting the analysis algorithm to a certain degree of modulation is crucial and could be balanced with a multi resolution method of some sort, for example a wavelet transform based one, a similar idea already pointed out in [15].

6. ACKNOWLEDGEMENTS

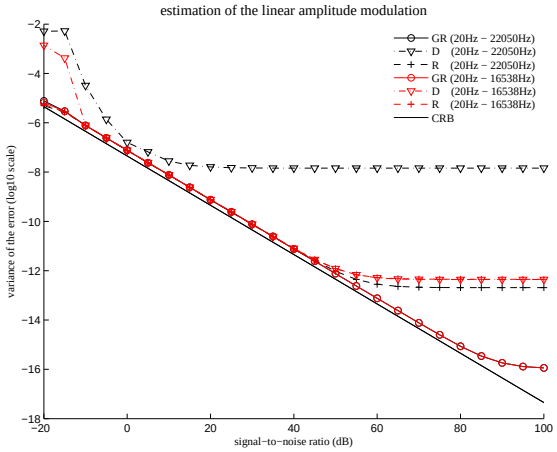
The authors wish to thank Sylvain Marchand for his help with providing original code for reassignment, the derivative method and the testing environment. Research was funded by 'Slovene human resources development and scholarship fund' ('Javni sklad Republike Slovenije za razvoj kadrov in štipendije').



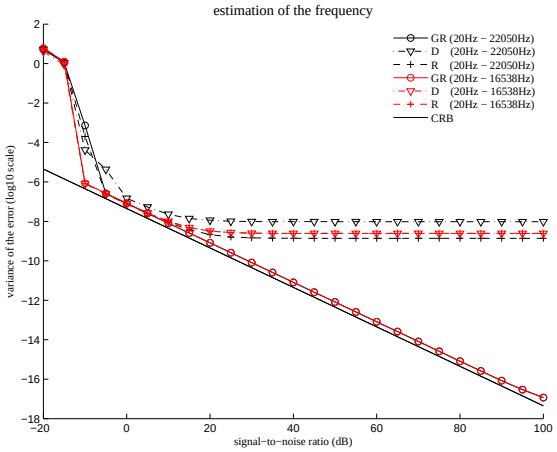
(a) Amplitude estimation error for linear log-AM/linear FM sinusoid in comparison with CRB



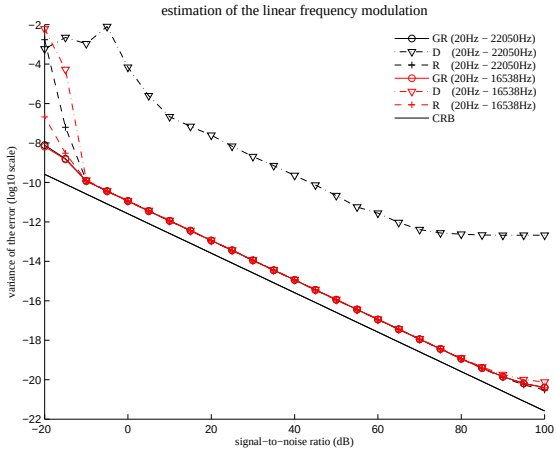
(b) Phase estimation error for linear log-AM/linear FM sinusoid in comparison with CRB



(c) Log-AM estimation error for linear log-AM/linear FM sinusoid in comparison with CRB



(d) Frequency estimation error for linear log-AM/linear FM sinusoid in comparison with CRB



(e) FM estimation error for linear log-AM/linear FM sinusoid in comparison with CRB

Figure 1: Parameter estimation errors for reassignment, the derivative method and generalized reassignment.

7. APPENDIX A

As already briefly mentioned in section 1, the equality ${}^{\mathrm{R}}S_w = {}^{\mathrm{D}}S_w$ holds for all t and ω, yet each equation yields a different time derivative expression. When the window function is assumed to be of finite support, and consequently the infinite integral bounds are replaced by finite ones, the expressions for ${}^{\mathrm{R}}S_w$ and ${}^{\mathrm{D}}S_w$ change to:

$$ {}^{\mathrm{D}}S_w(t,\omega) = \int_{-\frac{T}{2}}^{\frac{T}{2}} s(\tau+t)w(\tau)e^{-j\omega\tau}d\tau \tag{36} $$

$$ {}^{\mathrm{R}}S_w(t,\omega) = \int_{t-\frac{T}{2}}^{t+\frac{T}{2}} s(\bar{\tau})w(\bar{\tau}-t)e^{-j\omega(\bar{\tau}-t)}d\bar{\tau}, \tag{37} $$

where the window function time support is assumed to be T. The partial time derivative of integral expression 37 should be taken with care, as integral limits depend on time and the time derivative operator cannot simply be brought inside the integral. Using the Leibniz's integral rule on ${}^{\mathrm{R}}S_w$ yields:

$$ \begin{aligned} &\frac{\partial}{\partial t}{}^{\mathrm{R}}S_w(t,\omega) = \\ &\frac{\partial}{\partial t}\int_{t-\frac{T}{2}}^{t+\frac{T}{2}} s(\bar{\tau})w(\bar{\tau}-t)e^{-j\omega(\bar{\tau}-t)}d\bar{\tau} = \\ &s(t+\frac{T}{2})\overbrace{w(\frac{T}{2})}^{=0}e^{-j\omega\frac{T}{2}} - s(t-\frac{T}{2})\overbrace{w(-\frac{T}{2})}^{=0}e^{j\omega\frac{T}{2}} + \\ &\int_{t-\frac{T}{2}}^{t+\frac{T}{2}} s(\bar{\tau})\frac{\partial}{\partial t}\left[w(\bar{\tau}-t)e^{-j\omega(\bar{\tau}-t)}\right]d\bar{\tau}. \end{aligned} \tag{38} $$

With the additional restriction that the window function reaches 0 at both its edges, e.g.: $w(\frac{T}{2}) = w(-\frac{T}{2}) = 0$, the first two terms of the above expression can be neglected. The restriction modifies equation 38 in a way, as if the integral boundaries wouldn't depend on the time variable and thus the time derivative operator can simply be brought inside the integral. Throughout this appendix it will be assumed, that the window function and its arbitrary time derivative reach 0 at both edges, eg: $w^{(k)}(\frac{T}{2}) = w^{(k)}(-\frac{T}{2}) = 0$ for all k and consequently the time derivation operator can always be brought inside the integral. Note, that the partial frequency derivative operators can be brought inside the integral for both ${}^{\mathrm{R}}S_w$ and ${}^{\mathrm{D}}S_w$, as well as the partial time derivative in the case of ${}^{\mathrm{D}}S_w$ without any additional restrictions. In the following derivations the integral bounds will be omitted for the sake of clarity. For reassignment the following holds:

$$ \begin{aligned} &\frac{\partial}{\partial t}{}^{\mathrm{R}}S_w = \int s(\tau)\frac{\partial}{\partial t}\left[w(\tau-t)e^{-j\omega(\tau-t)}\right]d\tau = \\ &\int s(\tau)\overbrace{\frac{\partial}{\partial t}[w(\tau-t)]}^{-w'(t)}e^{-j\omega(\tau-t)}d\tau + \\ &\int s(\tau)w(\tau-t)\overbrace{\frac{\partial}{\partial t}\left[e^{-j\omega(\tau-t)}\right]}^{j\omega e^{-j\omega(\tau-t)}}d\tau = \\ &-S_{w'} + j\omega S_w \end{aligned} \tag{39} $$

$$ \frac{\partial}{\partial\omega}{}^{\mathrm{R}}S_w = \int s(\tau)w(\tau-t)\overbrace{\frac{\partial}{\partial\omega}\left[e^{-j\omega(\tau-t)}\right]}^{-j(\tau-t)e^{-j\omega(\tau-t)}}d\tau = -jS_{tw} \tag{40} $$

$$ \begin{aligned} &\frac{\partial}{\partial t}{}^{\mathrm{R}}S_{tw} = \\ &\int s(\tau)\frac{\partial}{\partial t}\left[w(\tau-t)(\tau-t)e^{-j\omega(\tau-t)}\right]d\tau = \\ &\int s(\tau)\overbrace{\frac{\partial}{\partial t}[w(\tau-t)]}^{-w'(\tau-t)}(\tau-t)e^{-j\omega(\tau-t)}d\tau + \\ &\int s(\tau)w(\tau-t)\overbrace{\frac{\partial}{\partial t}[\tau-t]}^{-1}e^{-j\omega(\tau-t)}d\tau + \\ &\int s(\tau)w(\tau-t)(\tau-t)\overbrace{\frac{\partial}{\partial t}\left[e^{-j\omega(\tau-t)}\right]}^{j\omega e^{-j\omega(\tau-t)}}d\tau = \\ &-S_{w'} - S_w + j\omega S_w. \end{aligned} \tag{41} $$

It is informative to generalize the time and frequency derivative expressions for the reassignment STFT using arbitrary window time derivative, ramped with an arbitrary polynomial:

$$ \begin{aligned} &\frac{\partial}{\partial t}{}^{\mathrm{R}}S_{t^n w^{(k)}} = \\ &\int s(\tau)\frac{\partial}{\partial t}\left[w^{(k)}(\tau-t)(\tau-t)^n e^{-j\omega(\tau-t)}\right]d\tau = \\ &\int s(\tau)\overbrace{\frac{\partial}{\partial t}[w^{(k)}(\tau-t)]}^{-w^{(k+1)}(\tau-t)}(\tau-t)^n e^{-j\omega(\tau-t)}d\tau + \\ &\int s(\tau)w^{(k)}(\tau-t)\overbrace{\frac{\partial}{\partial t}[(\tau-t)^n]}^{-n(\tau-t)^{n-1}}e^{-j\omega(\tau-t)}d\tau + \\ &\int s(\tau)w^{(k)}(\tau-t)(\tau-t)^n\overbrace{\frac{\partial}{\partial t}\left[e^{-j\omega(\tau-t)}\right]}^{j\omega e^{-j\omega(\tau-t)}}d\tau = \\ &-S_{t^n w^{(k+1)}} - S_{t^{n-1}w^{(k)}}n + j\omega S_{t^n w^{(k)}} \end{aligned} \tag{42} $$

$$\frac{\partial}{\partial\omega}{}^{\mathsf{R}}S_{t^n w^{(k)}} = \int s(\tau)w^{(k)}(\tau-t)(\tau-t)^n \overbrace{\frac{\partial}{\partial\omega}\left[e^{-j\omega(\tau-t)}\right]}^{-j(\tau-t)e^{-j\omega(\tau-t)}} d\tau = -jS_{t^{n+1}w^{(k)}}. \quad (43)$$

Equation 42 corresponds to equations 39 and 41 for the values of $k=0, n=0$ and $k=0, n=1$ respectively and equation 43 corresponds to equation 40 for the values of $k=0, n=0$.

Analogously, for the derivative method STFTs, time and frequency derivatives can be generalized for an arbitray signal time derivative, using a window ramped with an arbitrary polynomial:

$$\frac{\partial}{\partial t}{}^{\mathsf{D}}S^{(k)}_{t^n w} = \int \frac{\partial}{\partial t}\left[s^{(k)}(\tau+t)\right] w(\tau)\tau^n e^{-j\omega\tau}d\tau = S^{(k+1)}_{t^n w} \quad (44)$$

$$\frac{\partial}{\partial\omega}{}^{\mathsf{D}}S^{(k)}_{t^n w} = \int_{-\infty}^{\infty} s(\tau+t)w(\tau)\tau^n \overbrace{\frac{\partial}{\partial\omega}\left[e^{-j\omega\tau}\right]}^{-j\tau e^{-j\omega\tau}} d\tau = -jS^{(k)}_{t^{n+1}w}. \quad (45)$$

8. APPENDIX B

Substituting the reassignment STFT expressions 16-19 into general equations for frequency, log-AM and FM estimations defined in 6,9,10 yields:

$$\begin{aligned}
{}^{\mathsf{R}}\hat{\omega}(t,\omega) &= \Im\left(\frac{-S_{w'}+j\omega S_w}{S_w}\right)\\
&= \Im\left(j\omega - \frac{S_{w'}}{S_w}\right)\\
&= \omega - \Im\left(\frac{S_{w'}}{S_w}\right)
\end{aligned} \quad (46)$$

$$\begin{aligned}
{}^{\mathsf{R}}\hat{\mu}(t,\omega) &= \Re\left(\frac{-S_{w'}+j\omega S_w}{S_w}\right)\\
&= \Re\left(j\omega - \frac{S_{w'}}{S_w}\right)\\
&= -\Re\left(\frac{S_{w'}}{S_w}\right)
\end{aligned} \quad (47)$$

$$\begin{aligned}
{}^{\mathsf{R}}\hat{\psi}(t,\omega) &= \frac{\partial^{\mathsf{R}}\hat{\omega}}{\partial t}\Big/\frac{\partial^{\mathsf{R}}\hat{t}}{\partial t}\\
\frac{\partial^{\mathsf{R}}\hat{\omega}}{\partial t} &= \Im\left(\frac{(S_{w''}-2j\omega S_{w'}-\omega^2 S_w)S_w}{(S_w)^2}\right)\\
&\quad - \Im\left(\frac{(-S_{w'}+j\omega S_w)^2}{(S_w)^2}\right)\\
&= \Im\left(\frac{S_{w''}S_w-(S_{w'})^2}{(S_w)^2}\right)
\end{aligned} \quad (48)$$

$$\begin{aligned}
\frac{\partial^{\mathsf{R}}\hat{t}}{\partial t} &= 1 - \Im\left(\frac{(-S_{w'}-S_w+j\omega S_w)S_w)}{(S_w)^2}\right)\\
&\quad - \Im\left(\frac{jS_w(-S_{w'}+j\omega S_w}{(S_w)^2}\right)\\
&= 1 - \Im\left(j\frac{(S_w)^2+S_{tw'}S_w-S_{tw}S_w}{(S_w)^2}\right)\\
&= \Re\left(\frac{S_{tw'}S_w-S_{tw}S_w}{(S_w)^2}\right).
\end{aligned} \quad (49)$$

Substituting the derivative method STFT expressions 20-23 into general equations for frequency and log-AM estimations defined in 6,9 yield straightforward expressions, however the expression for FM estimate 10 deserves more attention:

$$\begin{aligned}
{}^{\mathsf{D}}\hat{\psi}(t,\omega) &= \frac{\partial^{\mathsf{D}}\hat{\omega}}{\partial t}\Big/\frac{\partial^{\mathsf{D}}\hat{t}}{\partial t}\\
\frac{\partial^{\mathsf{D}}\hat{\omega}}{\partial t} &= \Im\left(\frac{S''_w S_w-(S'_w)^2}{(S_w)^2}\right)\\
&= \Im\left(\frac{S''_w}{S_w}\right) - \overbrace{\Im\left(\left(\frac{S'_w}{S_w}\right)^2\right)}^{2\Re\left(\frac{S'_w}{S_w}\right)\Im\left(\frac{S'_w}{S_w}\right)}\\
&= \Im\left(\frac{S''_w}{S_w}\right) - 2^{\mathsf{D}}\hat{\mu}(t,\omega)^{\mathsf{D}}\hat{\psi}(t,\omega)
\end{aligned} \quad (50)$$

$$\begin{aligned}
\frac{\partial^{\mathsf{D}}\hat{t}}{\partial t} &= 1 - \Im\left(\frac{-jS'_{tw}S_w+jS_{tw}S'_w}{(S_w)^2}\right)\\
&= 1 + \Re\left(\frac{S'_{tw}S_w-S_{tw}S'_w}{(S_w)^2}\right).
\end{aligned} \quad (51)$$

9. REFERENCES

[1] X. Serra, *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition*. PhD thesis, CCRMA, Department of Music, Stanford University, 1989.

[2] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, pp. 744–754, August 1986.

[3] G. Peeters and X. Rodet, "Sinola: A new analysis/synthesis method using spectrum peak shape distortion, phase and reassigned spectrum," in *Proc. of 25th International Computer Music Conference*, (Beijing, China), pp. 153–156, October 1999.

[4] M. Desainte-Catherine and S. Marchand, "High precision fourier analysis of sounds using signal derivatives," *Journal of the Audio Engineering Society*, vol. 48, no. 7/8, pp. 654–667, 1998.

[5] F. Keiler and S. Marchand, "Survey on extraction of sinusoids in stationary sounds," in *Proc. of 5th Int. Conference on Digital Audio Effects (DAFx-02)*, (Barcelona, Spain), September 2002.

[6] S. Marchand, "Improving spectral analysis precision with an enhanced phase vocoder using signal derivatives," in *Proc. of Digital Audio Effects Workshop (DAFX-98)*, (Barcelona, Spain), pp. 114–118, November 1998.

[7] B. David, G. Richard, M. Betser, and P. Collen, "Estimation of Frequency for AM/FM Models Using the Phase Vocoder Framework," *Signal Processing, IEEE Transactions on*, vol. 56, pp. 505–517, 2008.

[8] P. Depalle and S. Marchand, "Generalization of the derivative analysis method to non-stationary sinusoidal modeling," in *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx-08)*, (Espoo, Finland), September 2008.

[9] J. Smith and M. Abe, "AM/FM rate estimation for time-varying sinusoidal modeling," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, vol. 3, (Philadelphia, PA, USA), pp. 201–204, March 2005.

[10] A. Röbel, "Estimating partial frequency and frequency slope using reassignment operators," in *Proc. of the International Computer Music Conference (ICMC'02)*, (Gothenburg, Sweden), pp. 122–125, September 2002.

[11] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *Signal Processing, IEEE Transactions on*, vol. 43, no. 5, pp. 1068–1089, 1995.

[12] S. W. Hainsworth, *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, Department of Engineering, University of Cambridge, Cambridge, 2004.

[13] B. Hamilton, P. Depalle, and S. Marchand, "Theoretical and practical comparisons of the reassignment method and the derivative method for the estimation of the frequency slope," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, (New Paltz, NY), October 2009.

[14] W. Xau and M. Sandler, "Notes on model-based non-stationary sinusoid estimation methods using derivative," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, (Como, Italy), September 2009.

[15] M. Betser, "Sinusoidal polynomial parameter estimation using the distribution derivative," *Signal Processing, IEEE Transactions on*, vol. 57, pp. 4633–4645, December 2009.

[16] F. Auger and P. Flandrin, "Generalization of the reassignment method to all bilinear time-frequency and time-scale representations," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '94)*, vol. iv, (Washington, DC, USA), pp. 317–320, 1994.

PHASESHAPING OSCILLATOR ALGORITHMS FOR MUSICAL SOUND SYNTHESIS

Jari Kleimola[1], **Victor Lazzarini**[2], **Joseph Timoney**[2], **and Vesa Välimäki**[1]

[1] Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland

[2] Sound and Digital Music Technology Group, National University of Ireland, Maynooth, Ireland

jari.kleimola@tkk.fi, victor.lazzarini@nuim.ie, jtimoney@cs.nuim.ie, vesa.valimaki@tkk.fi

ABSTRACT

This paper focuses on phaseshaping techniques and their relation to classical abstract synthesis methods. Elementary polynomial and geometric phaseshapers, such as those based on the modulo operation and linear transformations, are investigated. They are then applied to the generation of classic and novel oscillator effects by using nested phaseshaping compositions. New oscillator algorithms introduced in this paper include single-oscillator hard sync, triangle modulation, efficient supersaw simulation, and sinusoidal waveshape modulation effects. The digital waveforms produced with phaseshaping techniques are generally discontinuous, which leads to aliasing artifacts. Aliasing can be effectively reduced by modifying samples around each discontinuity using the previously proposed polynomial bandlimited step function (polyBLEP) method.

1. INTRODUCTION

The generation of complex musical timbres has been approached from various angles in sound computing. One elegant solution, which has provided a wide scope for research and implementation, has been that of distortion techniques. Within this area, various techniques have been put forward, such as frequency modulation (FM) [3], phase distortion (PD) [5,9], nonlinear waveshaping [1,10,14,16], and discrete summation formulae (DSF) [11]. These are in many cases equivalent and can be used as alternative ways to describe and implement a given algorithm, as discussed in [7].

In particular, the waveshaping method provides a computationally simple means to produce potentially rich spectra. Its principle is quite straightforward, starting with a discrete-time sinusoidal signal,

$$x(n) = \sin(\omega n), \tag{1}$$

where ω is the angular frequency and n is the discrete sample index, a complex (i.e., non-sinusoidal) spectrum can be obtained via a mapping such as

$$y(n) = f[x(n)], \tag{2}$$

where $f[.]$ is an arbitrary nonlinear function called a *waveshaper*. The well-known classic FM synthesis equation, for instance, can be rewritten as a waveshaping expression

$$\begin{aligned}&\cos[\omega_c n + I x(n)] = \\ &\cos(\omega_c n)\cos[I x(n)] - \sin(\omega_c n)\sin[I x(n)],\end{aligned} \tag{3}$$

where ω_c is the carrier frequency and the waveshapers cos(.) and sin(.) act on the sinusoidal modulation signal $x(n)$ of Equation 1.

Similarly, it is possible to describe PD as a form of waveshaping. This is demonstrated by starting with the following expression defining a sinusoidal oscillator:

$$y(n) = \cos[2\pi\phi(n)]. \tag{4}$$

The function $\phi(n)$ is the normalized phase defined by

$$\phi(n) = [\phi(n-1) + f_0 / f_s] \bmod 1, \tag{5}$$

where f_0 is the fundamental frequency, f_s is the sampling rate, and x mod 1 = $x - \lfloor x \rfloor$, and $\lfloor x \rfloor$ is the floor function denoting the largest integer that is not greater than x. To implement a PD oscillator, the phase is then applied to a nonlinear function $g(x)$:

$$y(n) = \cos\{2\pi\, g[\phi(n)]\}. \tag{6}$$

A linear $g(x)$ would result in a sinusoid whose frequency is transposed. However, with nonlinear $g(x)$, the shape of the output waveform is modified.

From a waveshaping perspective, Equation 4 can be described as a sinusoidal waveshaper acting on a complex input waveform $s(n) = 2\pi\phi(n)$:

$$y(n) = f[s(n)]. \tag{7}$$

This transforms the phase signal $s(n)$ into the output signal $y(n)$ by means of a waveshaper $f(x) = \cos(x)$. The waveshaper can be implemented as a function or as a lookup table that acts on the normalized value of the phase signal. The typical phase generator producing $s(n)$ is the unipolar modulo counter $\phi(n)$ of Equation 5, which is also a unipolar geometric non-bandlimited sawtooth wave.

In this vein, Equation 6 can be seen as based on a form of double waveshaping, where two functions, $g(x)$ and cos(x), are applied to an input sawtooth wave $\phi(n)$. This is perfectly equivalent to the principle of distorting the phase function $\phi(n)$ of a sinusoidal oscillator.

In this paper, the term 'phaseshaping' [7] is used to describe the generalization of the phase function distortion $g[\phi(n)]$. The aim here is to investigate elementary polynomial and geometrical phaseshapers, and then discuss their application in classic and novel oscillator algorithms.

2. ELEMENTARY PHASESHAPERS

The investigation is began by proposing two fundamental phaseshaping concepts, entitled nested phaseshaping and phaseshaper entities. *Nested phaseshaping* is related to function composition, in which the result of the inner function serves as the input to the outer function. Equation 8 shows an example of nesting at three levels, expressed in the basic form in Equation 8a and its equivalent shorthand notation in Equation 8b.

$$y(n) = f\{g[h(x)]\}, \tag{8a}$$

$$y(n) = f \circ g \circ h(x). \tag{8b}$$

For the purposes of this paper, x is assumed to be a signal which flows from the inner function towards the outer ones, transforming at each step into the final shape given by the outmost function. The graphical representation of this composition is thus a signal block diagram, similar to the one shown in Figure 1.

Figure 1. Graphical representation of Equation 8.

It is further assumed that the source of the chain is the unipolar modulo counter $\phi(n)$ of Equation 5, and that the rightmost block is the waveshaper producing the final output signal. The blocks or functions between these two extremes are called phaseshapers, because they act on the phase signal $\phi(n)$ and because the input of the final waveshaper is essentially a phase signal as well. Having said this, note that in some cases the output of the chain is the phase signal itself instead of the product of the waveshaper.

Phaseshaper entities are frequently used phaseshapers that have fixed predefined semantics. These include

$$\mathrm{mod}_1[x(n)] = x(n) \bmod 1 \tag{9a}$$

$$\mathrm{mod}_m[x(n), m] = x(n) \bmod m \tag{9b}$$

$$g_\mathrm{b}[x(n)] = 2x(n) - 1 \tag{9c}$$

$$g_\mathrm{u}[x(n)] = 0.5x(n) + 0.5, \tag{9d}$$

where mod_1 is the modulo-1 operation, mod_m is the real-valued modulo-m operation ($m \in$ R), g_b is the bipolar transformation converting a unipolar signal into its bipolar form, and g_u its opposite unipolar transformation.

2.1 Ramp-like Fractional Period Phase Signals

Phaseshaper entities g_b and g_u are linear transformations, whose general expression is given by the phaseshaper

$$g_\mathrm{lin}[x(n), a_{1,0}] = a_1 x(n) + a_0\ , \tag{10}$$

where a_1 and a_0 are the scaling and shifting factors, respectively. Assuming that $x(n)$ is given by $\phi(n)$ – which is restricted to values between 0 and 1 – one notices that the output of g_lin is no longer constrained to the range [0,1], which is the expected normalized phase range of most waveshaper terminals of the shaper chain.

The output of g_lin should therefore be normalized. One way of doing this is to apply the mod_1 phaseshaper entity to obtain

$$y(n) = \mathrm{mod}_1 \circ g_\mathrm{lin}[x(n), a_{1,0}]. \tag{11}$$

The effect of this normalization is seen in Figure 2, which plots the output of Equation 11 using parameter values $a_1 = 1.5$ and $a_0 = 0$. The sampling rate $f_\mathrm{s} = 44.1$ kHz is used in all examples of this paper. In this example, the modulo operation is activated first within the context of mod_1 (producing the full-height phase cycle) and then within the context of g_lin (producing the fractional phase cycle)[1]. Parameter a_1 thus controls the length of the phase period (when $a_1 > 1$) or the slope of the phase signal (when $a_1 < 1$). The shifting term a_0 contributes to the DC offset of the produced phase signal.

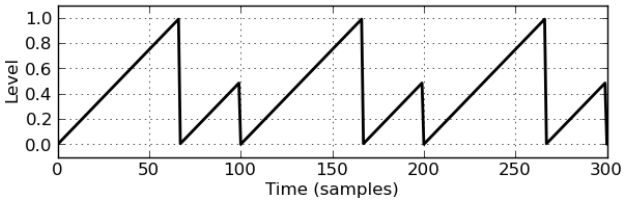


Figure 2. Ramp-like phase signal with a fractional phase period ($a_1 = 1.5$ and $a_0 = 0$). The fundamental frequency is $f_0 = 441$ Hz, as in all plots of this section.

2.2 Triangular Fractional Period Phase Signals

The unipolar modulo counter signal $\phi(n)$ can be transformed into a bipolar sawtooth waveform by applying the g_b phaseshaper entity of Equation 9c. Then, by feeding this sawtooth waveform through the absolute value function, a unipolar triangular signal [20] is obtained, which can be further shaped by g_lin and mod_1 to get phaseshaper

$$g_\mathrm{tri}[x(n), a_{1,0}] = \mathrm{mod}_1 \circ g_\mathrm{lin} \circ \mathrm{abs}\{g_\mathrm{b}[x(n)]\}\ . \tag{12}$$

Alternatively the abs{.} term of Equation 12 can be replaced with the piecewise linear triangular waveform definition

$$s_\mathrm{tri}(x) = \begin{cases} 2x, & \text{when } 0 \le x < 0.5, \\ 2 - 2x, & \text{when } 0.5 \le x \le 1. \end{cases} \tag{13}$$

The fractional period phase signal produced by g_tri is depicted in Figure 3, which shows that because the slope of the triangle wave is two times steeper than that of a

[1] Here the term *phase cycle* is adapted to describe the segment that takes the phase value from 0 to 1, and the term *phase period* to describe the total period of the modulo counter signal $\phi(n)$.

sawtooth, the frequency of the phase cycle is doubled. The phase period of g_{tri} therefore contains two complete periods of Equation 11 and, as expected, the latter period is reversed in time. Because of this symmetry, g_{tri} produces less dramatic effects on the output of the shaper chain.

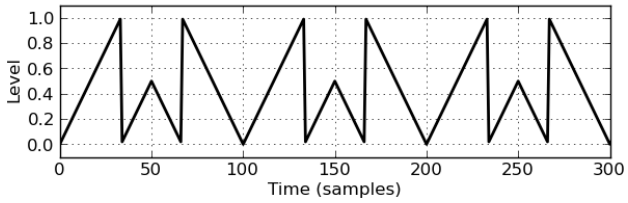


Figure 3. Triangular phase signal with a fractional phase period ($a_1 = 1.5$ and $a_0 = 0$).

2.3 Rectangular Signals

The unipolar modulo counter signal $\phi(n)$ can also be shaped into a unipolar square wave by first replacing the abs{.} term of g_{tri} with the signum function and then applying the unipolar transformation entity g_{u} to the result. Unfortunately, this construction does not allow for variable-width duty cycles.

Variable-width pulse signals can be generated by subtracting two out-of-phase ramp signals from each other [19], and then by offsetting the difference with the duty width, it is possible to obtain their unipolar representations. The generating phaseshaper is given by

$$g_{\text{pulse}}[x(n), w] = x(n) - x(n + wP) + w, \tag{14}$$

where w defines the pulse width ($0 \le w \le 1$) and $P = f_s/f_0$ is the period of $x(n)$. Since Equation 14 is linear and does not thus introduce aliasing, it is well suited for situations where $x(n)$ is a bandlimited or an antialiased signal. However, if aliasing problems are not a concern, the trivial unipolar pulse waveform definition

$$s_{\text{pulse}}(x) = \begin{cases} 1, & \text{when } 0 \le x < w \\ 0, & \text{when } w \le x \le 1 \end{cases} \tag{15}$$

is able to produce similar results more efficiently.

Although a rectangular signal is not a useful phase signal by itself, it may be combined with other phaseshapers for two-segment phase sequences. For instance, the expression for the variable-slope phase signal of Figure 4 is $x(n)\{1 + g_{\text{pulse}}[x(n) - 1, w]\}$. Another application for rectangular signals is the algebraic sawtooth shifter described in [4].

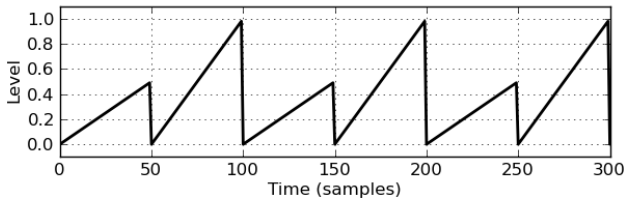


Figure 4. Variable-slope phase signal ($w = 0.5$).

2.4 Tilted Triangular Fractional Period Phase Signals

Instead of subtracting two sawtooth waveforms from each other, subtracting two out-of-phase parabolic waveforms produces a variable-slope triangle wave [13]. Using phaseshaping techniques, this can be implemented as

$$s_{\text{vtri}}[x(n), w] = a_{\text{T}}\left\{ g_{\text{b}}{}^2[x(n)] - g_{\text{b}}{}^2[x(n - w)] \right\} + b_{\text{T}}, \tag{16}$$

where w is the duty width, $a_{\text{T}} = 1/[8(w - w^2)]$, and $b_{\text{T}} = 0.5$. Although Equation 16 may be used as a standalone phase generator, it can be further generalized by shaping it with a g_{lin} and mod_1 sequence. This results in the phaseshaper

$$g_{\text{vtri}}[x(n), w, a_{1,0}] = \text{mod}_1 \circ g_{\text{lin}} \circ s_{\text{vtri}}[x(n)]. \tag{17}$$

Figure 5 plots a fractional period phase signal generated by g_{vtri}. Comparing this with Figure 3, it is noted that the slopes of the up- and down-ramp cycles are weighted by the duty width w. As expected, with $w = 0.5$ the slopes become equal in magnitude, at which point g_{vtri} and g_{tri} produce identical results. Therefore, Equation 17 can be seen as a generalization of Equation 12.

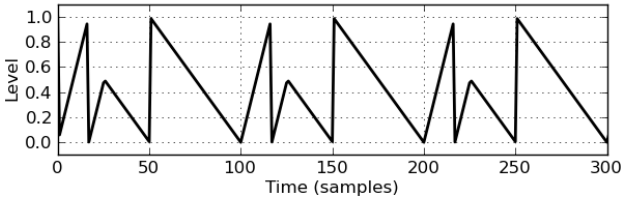


Figure 5. Variable-slope triangular phase signal with a fractional phase period ($a_1 = 1.5$, $a_0 = 0$, $w = 0.75$).

2.5 Phase Signals with Ripples

The definition of the general modulo operation of Equation 9b is

$$x(n) \bmod m = x(n) - m\lfloor x(n)/m \rfloor, \tag{18}$$

where $m \in \mathbb{R}$ is the real-valued wrapping modulus. For efficiency reasons, practical applications usually set $m = 1$, making Equation 18 equal to the fractional part of $x(n)$. In some applications, however, it is desirable to generate a phaseshaper whose output is decorated with small-amplitude ripples. This can be achieved by utilizing the phaseshaper entity mod_m (with a low fractional m value), as in

$$g_{\text{ripple}}[x(n), m] = x(n) + \text{mod}_m[x(n), m]. \tag{19}$$

An example phase signal generated by this phaseshaper is shown in Figure 6.

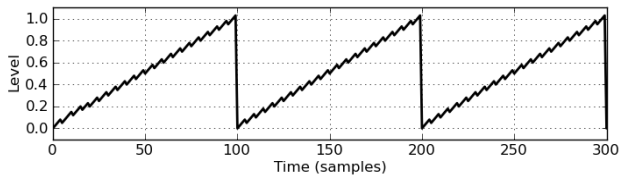


Figure 6. Phase signal with ripples ($m = 0.05$).

3. OSCILLATOR ALGORITHMS

This section describes the application of the elementary phaseshapers in classic and novel oscillator algorithms.

3.1 Waveslices

The waveforms produced by physical analog oscillators diverge from trivial piecewise linear sawtooth, pulse, and triangle waveshapes. Although these deviations are subtle in the spectral domain, they contribute to the characteristic sound of the synthesizer [15].

These nonlinear waveshapes may be approximated with higher order polynomial or sinusoidal waveshapers. For example, Figure 7a shows an approximation of the Minimoog Voyager sawtooth waveform, which was generated using

$$y(n) = g_{\mathrm{b}} \circ \sin\left\{2\pi\, g_{\mathrm{lin}}\left[\phi(n), a_1 = 0.25\right]\right\}. \qquad (20)$$

Parameter a_1 is set to a value smaller than unity so that only a portion of the entire sine wave period is included in the output. The spectrum of the waveform produced by Equation 20 is shown in Figure 7b. As can be seen, the abrupt transition caused by the modulo operation of $\phi(n)$ introduces a questionable amount of aliasing.

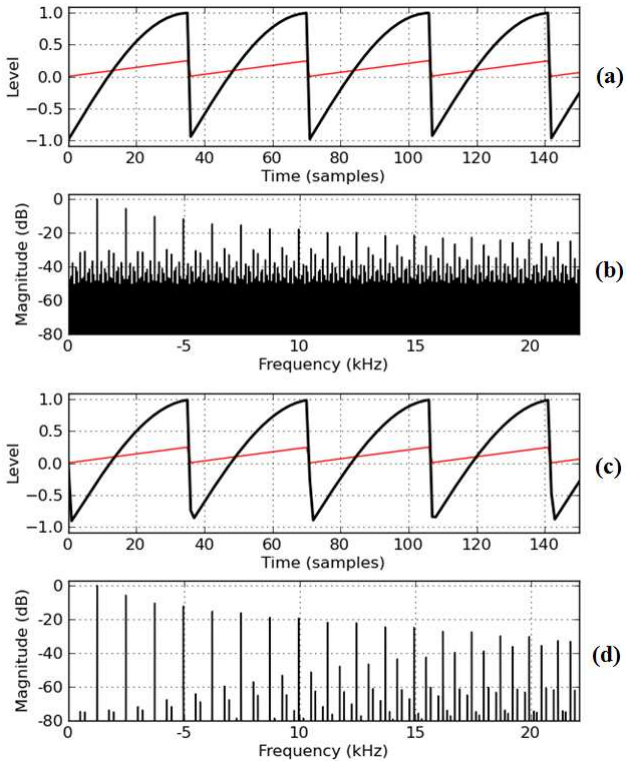


Figure 7. Approximation of the Minimoog Voyager sawtooth waveform. (a,b) Trivial and (c,d) aliasing-suppressed implementation. The thin lines of (a) and (c) plot the phase signal, while the thick lines show the waveshaper output (f_0 = 1245 Hz).

3.2 Antialiasing

The amount of aliasing can be suppressed by smoothing the transition in the time domain. An efficient method to accomplish this is the polynomial bandlimited step function (polyBLEP) [18], which is a simplification of the minBLEP method originally proposed by Brandt [2]. PolyBLEP modifies the values of two samples that are located before and after the modulo transition by evaluating a second-order correction polynomial and adding the result to the values of the two original waveform samples.

Figures 7c and 7d show the aliasing-suppressed waveform and spectrum of Equation 20 after applying the polyBLEP method. The aliasing is suppressed considerably at low and middle frequencies and, although the artifacts are still clearly visible in the spectrum plot, their effect is greatly diminished because of the properties of human hearing. The effect of transition smoothing is also visible in the time domain as the minima of the waveform do not reach the level of –1. Interestingly, the same effect is also observable in the original analog Minimoog Voyager waveform.

This suggests yet another phaseshaper entity that applies the polyBLEP method to its input signal, thereby performing a soft modulo-1 operation. This antialiasing phaseshaper is denoted as

$$\mathrm{mod}_{\mathrm{s}}\left[x(n), T_{\mathrm{s}}, h\right] = \mathrm{polyBLEP}\left[x(n), T_{\mathrm{s}}, h\right], \qquad (21)$$

where $T_{\mathrm{s}} = f_0 / f_{\mathrm{s}}$ is the phase increment of signal $x(n)$ and h is the maximum height of the discontinuity. The sign of h should be negative for falling transitions. A detailed explanation of the polyBLEP is out of the scope of this paper, but interested readers may consult Reference [18] and the source code published in the companion page of this paper[2].

3.3 Oscillator Synchronization

In classic oscillator hard synchronization (hardsync), the phase of the slave oscillator is reset each time the master oscillator finishes its cycle [2,17]. As shown in Figure 2, modulo-based phaseshaping is capable of producing similar effects by first utilizing the linear transformation phaseshaper g_{lin} and then processing the result with the modulo-1 phaseshaper entity mod_1. The latter operation synthesizes the free-running cycles of the slave oscillator, while the former generates the hardsynced transition. A computationally efficient trivial single-oscillator hardsync implementation is therefore given by the phaseshaping composition

$$y(n) = g_{\mathrm{b}} \circ \mathrm{mod}_1 \circ g_{\mathrm{lin}}\left[x(n), a_1\right] = 2\left[a_1 x(n) \bmod 1\right] - 1. \qquad (22)$$

The synchronization rate between the master and the slave oscillator is modeled by a_1, which is given in terms of the classic hardsync implementation as

$$a_1 = f_{\mathrm{slave}} / f_{\mathrm{master}}, \qquad (23)$$

where f_{slave} is the slave and f_{master} is the master oscillator frequency, respectively. Figure 8 shows the waveform and spectrum produced by the aliasing-suppressed single-oscillator hardsync algorithm for a_1 = 2.5.

[2] http://www.acoustics.hut.fi/go/smc2010-phaseshaping

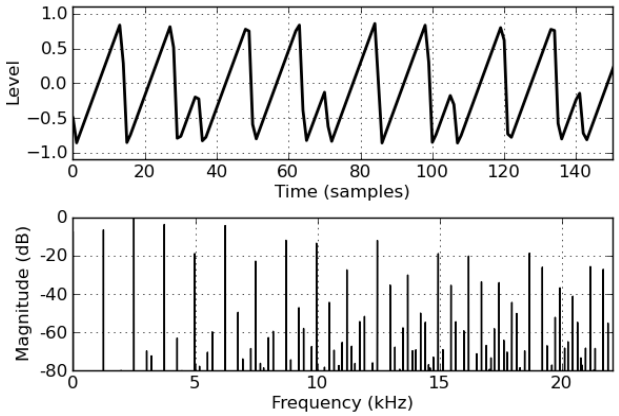


Figure 8. (top) Waveform and (bottom) spectrum of the single-oscillator hardsync algorithm in which the polyBLEP method is used to suppress aliasing (a_1 = 2.5, f_0 = 1245 Hz).

Instead of resetting the phase of the slave, oscillator soft synchronization (softsync) inverts the phase increment of the slave oscillator at the points of synchronization. The trivial single-oscillator softsync implementation utilizes the output of the phaseshaper g_{tri} of Equation 12 either directly or indirectly through a triangular waveshaper function $s_{\text{tri}}\{x\}$:

$$y(n) = g_{\text{b}} \circ g_{\text{tri}}[x(n), a_1] \tag{24a}$$

$$y(n) = g_{\text{b}} \circ s_{\text{tri}}\{g_{\text{tri}}[x(n), a_1]\}. \tag{24b}$$

Figure 9 shows the phase signal g_{tri} (thin line) on top of the resulting waveshaping operation of Equation 24b (thick line). The phase signal does not produce softsync in a strict sense, because the slopes of both ramps are inverted after the synchronization instant. However, this does not have a profound effect on the produced timbre.

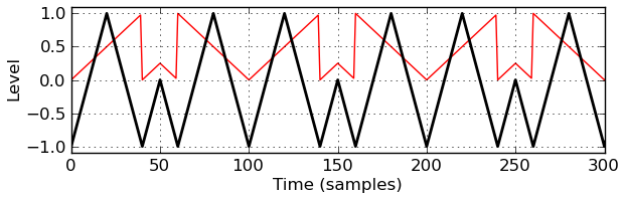


Figure 9. Trivial single-oscillator softsync effect. The thin line plots the phase signal, while the thick line shows the result of the waveshaping acting on that phase, as in all waveform plots in the subsequent examples (a_1 = 1.25 and f_0 = 441 Hz).

3.4 Pulse-width Modulation

Pulse-width modulation (PWM) changes the relative durations of the high and low state segments of a rectangular signal, while the frequency and the amplitude of the signal remain constant [17]. This can be achieved in two ways:

$$y(n) = g_{\text{b}} \circ g_{\text{pulse}}[x(n), w] \tag{25a}$$

$$y(n) = s_{\text{pulse,b}}\{\text{mod}_1 \circ g_{\text{lin}}[x(n), a_1]\}, \tag{25b}$$

where $s_{\text{pulse,b}}$ is the bipolar transformation of Equation 15. Both forms produce classic PWM when $0 < a_1 = w < 1$. When $a_1 > 1$, Equation 25b produces a trivial hardsynced square wave.

3.5 Triangle Modulation

One of the first commercial virtual analog synthesizers, the Roland JP-8000, introduced three original oscillator effects [15]. One of these effects is *triangle modulation* (see Figure 10a), which can be implemented using a scaled bipolar triangular phase signal $x_{\text{T}}(n)$ with a ceiling function:

$$\begin{aligned} x_{\text{T}}(n) &= a_{\text{TM}}\, g_{\text{tri}}[x(n), 2, -1] \\ y(n) &= 2\left(x_{\text{T}}(n) - \lceil x_{\text{T}}(n) - 0.5 \rceil\right), \end{aligned} \tag{26}$$

where a_{TM} is the modulation amount in the range [0.7, 1], and $\lceil x \rceil$ denotes the ceiling function, which returns the smallest integer not less than x. Figure 10b shows both signals of Equation 26 with a_{TM} = 0.82, corresponding to the Roland JP-8000 triangle modulation offset parameter value 64/127.

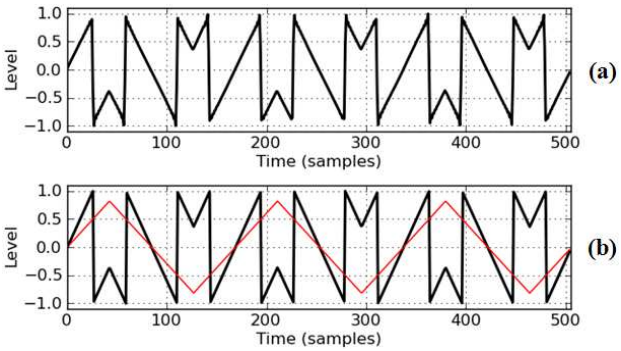


Figure 10. (a) Roland JP-8000 triangle modulation and (b) its simulation. The thin line plots the scaled phase signal $x_{\text{T}}(n)$, while the thick line shows the output signal $y(n)$ (a_{TM} = 0.82, f_0 = 261.63 Hz, and JP-8000 offset parameter = 64/127).

Higher amounts of modulation increase the slope of the ramp and the magnitude of the v-shaped segments. At the maximum modulation a_{TM} = 1 the magnitude of the v-shapes becomes 0.5. Figure 11 shows the effect of a_{TM} to the lower half of the baseband spectrum. As can be seen, the spectrum consists of odd harmonics only, the 3rd partial being the most prominent throughout the entire parameter range. The relative strengths of other harmonics change dynamically with a_{TM}, producing sweeping formant-like oscillator synchronization type effects.

The timbre that is produced by the maximum modulation amount a_{TM} = 1 can also be synthesized using the bitwise logical modulation [6]. This is not surprising, because the bitwise XOR operation is related to the staircase functions mod(.) and ceiling(.) employed here. The expression for the equivalent logical triangle modulation is

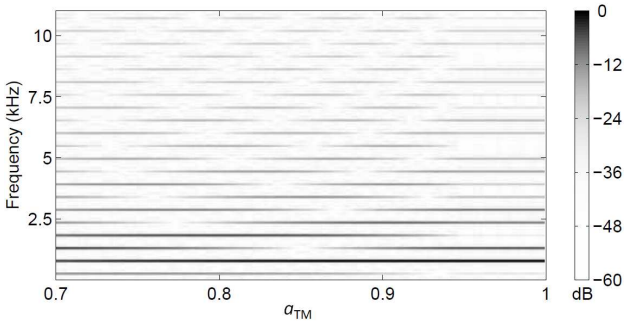


Figure 11. The effect of the modulation amount a_{TM} to the Roland JP-8000 triangle modulation spectrum.

$$y(n) = s_{\mathrm{tri}}\{2\, g_{\mathrm{tri}}[x(n)]\} \text{ xor } 0.5. \tag{27}$$

3.6 Supersaw

The most well-known Roland JP-8000 oscillator effect is *supersaw*, which emulates a bank of seven slightly detuned oscillators [15]. Previously, an algorithm for producing the supersaw signal using the bandlimited impulse-train method has been proposed in [12]. However, instead of utilizing seven oscillators, our supersaw simulation employs only one sinusoidal waveshaper that is driven by a slightly modified g_{ripple} phaseshaper:

$$y(n) = g_{\mathrm{b}} \circ \sin\{\mathrm{mod}_{m}[x(n), m_1] + \mathrm{mod}_{m}[x(n), m_2]\}, \tag{28}$$

where m_1 and m_2 are the ripple amounts, and $x(n) = g_{\mathrm{lin}}[\phi(n), a_1] = a_1\phi(n)$. The difference between the g_{ripple} phaseshaper of Equation 19 and that of Equation 28 is the added modulo operation of the first term.

Figure 12 shows three waveforms produced by the supersaw simulation algorithm, using three different ripple amounts m_1. Since $a_1 < 2\pi$, only a portion of the entire sine wave cycle is used as a virtual analog sawtooth oscillator. However, because $a_1 > 1$, the phase signal extends beyond a single phase cycle – thereby introducing an additional discontinuity to the ripple-edged waveform.

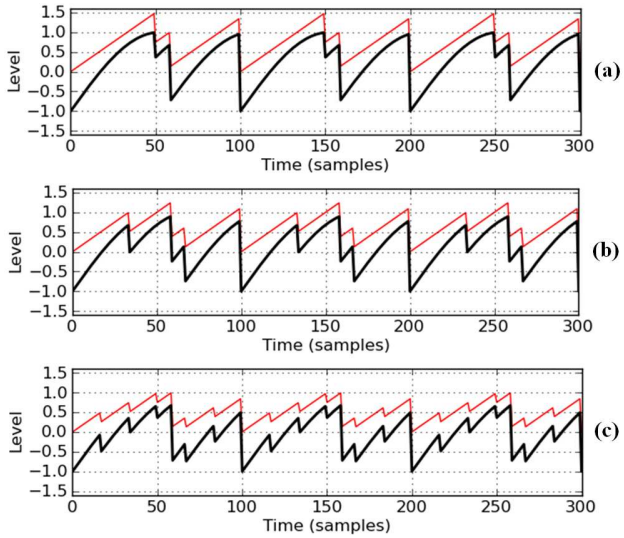


Figure 12. Supersaw simulation. (a) $m_1 = 0.75$, (b) $m_1 = 0.5$, (c) $m_1 = 0.25$ ($a_1 = 1.5$, $m_2 = 0.88$, and $f_0 = 441$ Hz).

Although Equation 28 is capable of synthesizing characteristic spectrally rich supersaw timbres, the sound is still not a convincing simulation of a multi-oscillator setup. This is due to a lack of timbral variations over time, which is a distinctive feature of a slightly detuned oscillator bank. To overcome this, a low frequency oscillator (LFO) may be connected to the m_1 parameter of the algorithm, as shown in Figure 13.

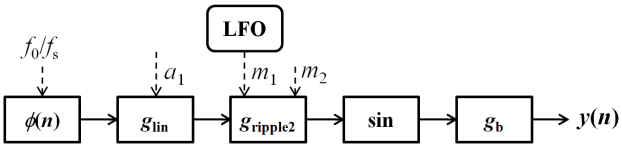


Figure 13. Block diagram of the supersaw simulation algorithm.

Figure 13 shows also that nested phaseshaping is a practical tool that provides a modular approach to sound synthesis and is therefore instantly applicable in systems such as Max, Pure Data, and Reaktor. However, some implementations might opt for minimizing the number of function calls in the code. An example of this is shown in Equation 22.

3.7 Phaseshaping for a Sinusoidal Waveshaper

3.7.1 Sinusoid with a Variable-slope Ramp Phase

Figure 14a shows the output of a sinusoidal waveshaper acting on the variable-slope phase signal of Figure 4. The waveshape consists of concatenated half- and full-cycle sine wave segments alternating at a frequency ratio of 1:2. The spectrum contains all harmonics and decays

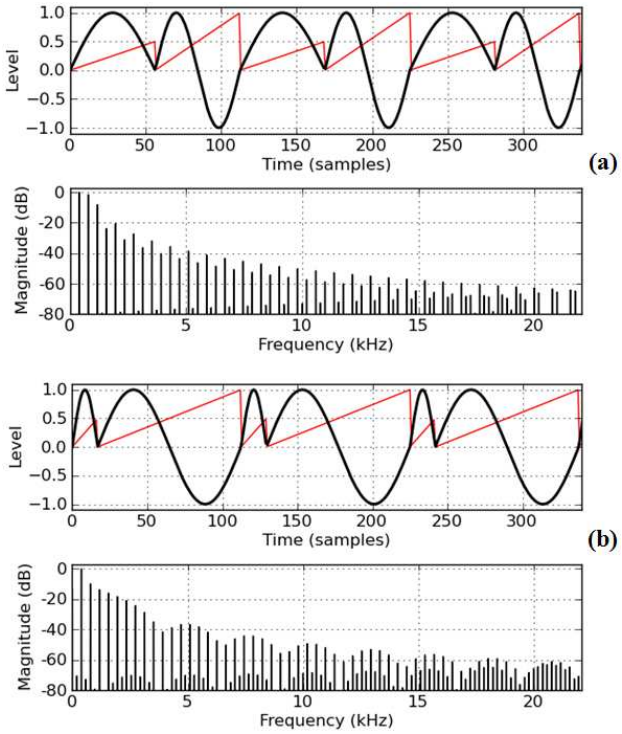


Figure 14. Variable-slope phase signal applied to a sinusoidal waveshaper. (a) Duty width $w = 0.50$, (b) duty width $w = 0.85$ ($f_0 = 392$ Hz).

fairly rapidly because the waveform has discontinuities only in its derivatives.

The phase signal of Figure 14a was generated by multiplying a ramp signal with a square waveform. By replacing the 50% duty-width square with a variable width pulse signal, it becomes possible to alter the relative widths of the half- and full-cycle sine segments, as shown in Figure 14b. As can be seen, the fundamental frequency component is reinforced as the width of the full-cycle segment is increased. The spectrum also shows modest formant regions that sweep across the baseband when the pulse width is modulated with an LFO.

3.7.2 Sinusoid with a Variable-slope Triangular Phase

The variable-slope triangular phase generator g_{vtri} of Equation 17 is closely related to the phase shape of the previous section. However, there are two major differences as can be seen in Figure 15. First, applying a sinusoidal waveshaper to the output of g_{vtri} produces a more prominent formant region, whose position may be controlled using the a_1 parameter. Second, outside this formant region, every fourth harmonic is missing from the spectrum. The aliasing artifacts are also more pronounced, because the symmetrical nature of the phaseshaper is reflected as the sharp peaks of the waveshaped output.

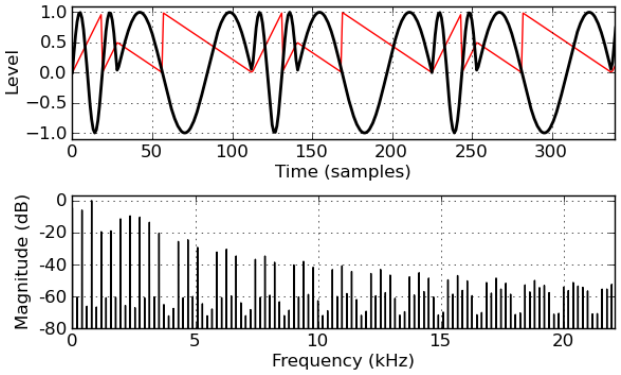


Figure 15. Variable-slope triangular phase signal applied to a sinusoidal waveshaper (a_1 = 1.5, w = 0.75, and f_0 = 392 Hz).

Decreasing the value of parameter a_1 below 1 bends the phase signal from a perfect triangle (a_1 = 0.5) towards a rising ramp shape (a_1 = 0). At a_1 = 0.5, the waveshaper output is a half-cycle sine wave, which gradually bends towards the extreme quarter-cycle segment shown in Figure 7. In between, the spectral tilt becomes less steep, thereby making it possible to control the amount of high end spectral content, as shown in Figure 16.

Lower values of a_1 produce more high end content, and at the same time, the amount of aliasing increases. By comparing Figure 16 to Figure 7, it is noted that polyBLEP provides better aliasing suppression than the sinusoidal waveshaping in effect here.

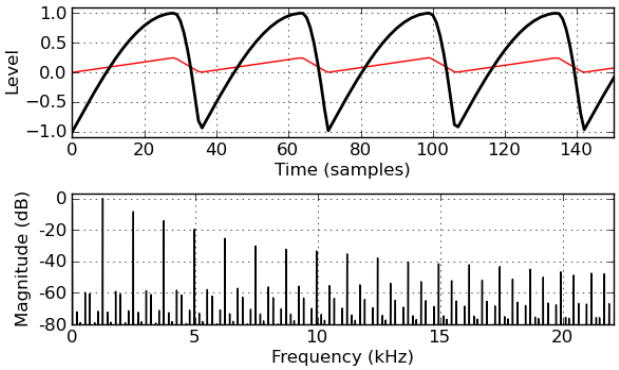


Figure 16. Bent sinusoidal half-cycle (a_1 = 0.25, w = 0.2, and f_0 = 1245 Hz).

4. CONCLUSIONS

This paper investigated elementary phaseshapers, which were based on low-level entities such as modulo operations and linear transformations. All elementary phaseshapers were derived from the unipolar modulo counter signal, which is a common building block of digital sound synthesis systems.

The elementary phaseshapers were then arranged into nested higher-level topologies to form polynomial and geometrical phaseshaper compositions. These included fractional period, variable-width and variable-slope ramp, triangular, rectangular, and ripple-edged phaseshapers.

The phaseshaper compositions were finally utilized in classic and novel oscillator effect algorithms. The novel algorithms comprised single-oscillator hardsync, triangle modulation, efficient supersaw simulation, and sinusoidal waveshape modulation effects.

These synthesis algorithms produce evolving spectra, which can be manipulated with a continuous controller device or a control rate function generator, using a compact set of synthesis parameters. The algorithms are most useful in providing animation to the otherwise static timbres, and as such, respond well to secondary control streams that carry minute articulated expressions of the performer.

Because of the modulo operation, the produced waveforms are generally discontinuous, leading to aliasing artifacts. However, it was found that a previously proposed polynomial bandlimited step function (polyBLEP) is an efficient method to reduce aliasing.

The authors believe that nested phaseshaping is a flexible tool that has many practical uses in the design and implementation of modular sound synthesis applications. Furthermore, because the phase signal has a profound effect on the produced timbre, phaseshaping may also be used in sculpting yet-unheard sonic material.

Online sound examples and software are available at http://www.acoustics.hut.fi/go/smc2010-phaseshaping.

5. ACKNOWLEDGMENTS

This work has been supported by the European Union as part of the 7th Framework Programme (SAME project,

ref. 215749) and by the Academy of Finland (project no. 122815).

6. REFERENCES

[1] D. Arfib: "Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves," *Journal of Audio Engineering Society*, Vol. 27, No. 10, pp. 757–768, 1979.

[2] E. Brandt: "Hard Sync Without Aliasing," *Proceedings of the International Computer Music Conference (ICMC 2001)*, Havana, Cuba, Sept. 17-22, 2001.

[3] J. Chowning: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *Journal of Audio Engineering Society*, Vol. 21, No. 7, pp. 526–534, 1973.

[4] B. Hutchins: "Analog Circuits for Sound Animation," *Journal of Audio Engineering Society*, Vol. 29, No. 11, pp. 814–820, 1981.

[5] M. Ishibashi: "Electronic Musical Instrument," *U.S. Patent 4,658,691*, 1987.

[6] J. Kleimola: "Audio Synthesis by Bitwise Logical Modulation," *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, pp. 67–70, 2008.

[7] V. Lazzarini and J. Timoney: "New Perspectives on Distortion Synthesis for Virtual Analog Oscillators," *Computer Music Journal*, Vol. 34, No. 1, pp. 28–40, 2010.

[8] V. Lazzarini, J. Timoney, and T. Lysaght: "Nonlinear Distortion Synthesis Using the Split-Sideband Method, with Applications to Adaptive Signal Processing," *Journal of Audio Engineering Society*, Vol. 56, No. 9, pp. 684–695, 2008.

[9] V. Lazzarini, J. Timoney, J. Pekonen, and V. Välimäki: "Adaptive Phase Distortion Synthesis," *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.

[10] M. Le Brun: "Digital Waveshaping Synthesis," *Journal of Audio Engineering Society*, Vol. 27, No. 4, pp. 250–266, 1979.

[11] J.A. Moorer: "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae," *Journal of Audio Engineering Society*, Vol. 24, No. 9, pp. 717–727, 1976.

[12] J. Nam, V. Välimäki, J.S. Abel, and J.O. Smith: "Efficient Antialiasing Oscillator Algorithms Using Low-order Fractional Delay Filters," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 4, pp. 773–785, 2010.

[13] M. Puckette: *The Theory and Technique of Electronic Music*, World Scientific Press, 2007.

[14] J.-C. Risset: "An Introductory Catalog of Computer-Synthesized Sounds," Bell Laboratories, 1969. Published as part of The Historical CD of Digital Sound Synthesis, *Computer Music Currents 13*, Wergo WER 20332, 1995.

[15] Roland: *JP-8000 Synthesizer Owner's Manual*, Roland Corporation, 1996.

[16] R.A. Schaefer: "Electronic Musical Tone Production by Nonlinear Waveshaping," *Journal of Audio Engineering Society*, Vol. 18, No. 4, pp. 413–417, 1970.

[17] A. Strange: *Electronic Music: Systems, Techniques and Controls*, William C Brown Pub., 1983.

[18] V. Välimäki and A. Huovilainen: "Antialiasing Oscillators in Subtractive Synthesis," *IEEE Signal Processing Magazine*, Vol. 24, No. 2, pp. 116–125, 2007.

[19] V. Välimäki and A. Huovilainen: "Oscillator and Filter Algorithms for Virtual Analog Synthesis," *Computer Music Journal*, Vol. 30, No. 2, pp. 19–31, 2006.

[20] V. Välimäki, J. Nam, J.O. Smith, and J.S. Abel: "Alias-Suppressed Oscillators Based on Differentiated Polynomial Waveforms," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 4, pp. 786–798, 2010.

Sparse Regression in Time-Frequency Representations of Complex Audio

Monika Dörfler, Gino Velasco
Nuhag, Faculty of Mathematics
University of Vienna, Austria
monika.doerfler@univie.ac.at
gino.velasco@univie.ac.at

Arthur Flexer, Volkmar Klien
Austrian Research Institute for Artificial Intelligence
arthur.flexer@ofai.at
volkmar.klien@ofai.at

ABSTRACT

Time-frequency representations are commonly used tools for the representation of audio and in particular music signals. From a theoretical point of view, these representations are linked to Gabor frames. Frame theory yields a convenient reconstruction method making post-processing unnecessary. Furthermore, using dual or tight frames in the reconstruction, we may resynthesize localized components from so-called sparse representation coefficients. Sparsity of coefficients is directly reinforced by the application of a ℓ^1-penalization term on the coefficients. We introduce an iterative algorithm leading to sparse coefficients and demonstrate the effect of using these coefficients in several examples. In particular, we are interested in the ability of a sparsity promoting approach to the task of separating components with overlapping analysis coefficients in the time-frequency domain. We also apply our approach to the problem of auditory scene description, i.e. source identification in a complex audio mixture.

1. INTRODUCTION

Time-frequency representations such as the spectrogram or short-time Fourier transform seem to be well suited for the representation of music. However, due to the uncertainty principle, a certain smearing of the time-frequency coefficients is unavoidable. This effect will often create overlap between components that would not be expected to overlap by their nature, e.g. two sinusoids with close frequencies. For other components, the overlapping area may be increased, thus complicating the task of separating certain components with approximately disjoint support in the time-frequency domain. For example, one might be interested in suppressing a certain instrument's contribution from a music signal. Such approaches are used in Computational Auditory Scene Analysis by the name of Time-Frequency masks. In this contribution, we describe the nature of time-frequency representations from a mathematical point of view. We then introduce a model and a corresponding algorithm for actively obtaining a sparse signal representation. The model rests on the fact that the time-frequency representations typically used in the audio signal processing community, are highly redundant. Hence, the representation coefficients are not unique and we may impose additional assumptions on the coefficients. Here, we will describe the effect of imposing an ℓ^1-penalization on the coefficients. We will show for several synthetic and real signals, that this leads to sharper representations and better separation properties. We apply the presented methods to the problem of auditory scene description. More precisely, given a mixture of known source sounds, we wish to determine the activity pattern for each source. This will be achieved by correlating representation coefficients of the sources with those of the mixture. In this setting, we compare the canonical time-frequency coefficients to those obtained from sparse regression in the time-frequency domain. The idea to use the sparse coefficients in place of the the canonical ones rests on the assumption, that these coefficients accurately capture the main characteristics of each of the sources. We will show that results obtained from sparse time-frequency representations improve those obtained from canonical representations. In particular, the amount of false positives is reduced, which is an important issue as pointed out in [1]. Thus, sparsity constraints help to avoid artificial correlations between signal components.

2. GABOR FRAMES: ANALYSIS AND SYNTHESIS

Given a discrete sequence of real or complex numbers, $x[n]$, $n \in \mathbb{Z}$, as well as a, usually compactly supported, window function $\varphi[n]$, $n \in \mathbb{Z}$, the short-time Fourier transform (or STFT) of $x[n]$ is given, for $k \in \mathbb{Z}$ and $\omega \in [-0.5, 0.5]$ by

$$\mathcal{V}_\varphi x(k,\omega) = \sum_{n=-\infty}^{\infty} x[n]\varphi[n-k]e^{-2\pi i\omega n}. \qquad (1)$$

Now, in practice, a subsampled version of (1) will usually be applied. Also, since the window φ has finite length l, we deal with a finite number of frequency bins. Hence, the result of the sampled STFT, also called Gabor transform, [2], is a matrix of size $N \times M$, where N is the number of time shifts by a time-constant, or hop-size, a considered. M is the number of frequency bins, hence the length of the FFT, given by l/b, b being the frequency-shift constant.
To gain a more general point of view, it is convenient, to consider the coefficients $\mathcal{V}_\varphi x(ka, mb)$ obtained from subsampling in (1), as inner products between the signal x and

time-frequency-shifted windows. We define the inner product x and $y \in \mathbb{C}^L$ as

$$\langle x, y \rangle = \sum_{n=0}^{L-1} x[n]\overline{y[n]}. \quad (2)$$

φ will from now on be called window function.

Definition 1 (Time-frequency shifts) *Let $x \in \mathbb{C}^L$.*
$T_k x[n] := x[n-k]$ is called translation operator or time shift by k.
$M_l x[n] := e^{\frac{2\pi i l n}{L}} x[n], l \in \mathbb{Z}$ is called modulation operator or frequency shift by l.
The composition of these operators, $M_l T_k$, is a time-frequency shift operator.

The family

$$\varphi_{k,m} := M_{mb} T_{ka} \varphi \quad (3)$$

for a window function $\varphi \in \mathbb{C}^L$, $m = 0, \ldots, M-1$ and $k = 0, \ldots, K-1$, where $Ka = Mb = L$, is called the set of *Gabor analysis functions*.
We next describe, under which conditions a signal is unambiguously defined by a family of Gabor atoms. The theory of *frames* gives the appropriate framework and we first state the defining inequalities for signals of finite energy.

Definition 2 *A set of Gabor analysis functions $\varphi_{k,m}$ in $\mathbb{C}^L$ is called a Gabor frame, if there exist constants $A, B > 0$, so that, for all $f \in \mathbb{C}^L$*

$$A\|f\|^2 \leq \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} |\langle f, \varphi_{k,m} \rangle|^2 \leq B\|f\|^2. \quad (4)$$

If $A = B$, then the functions $\varphi_{k,m}$ form a *tight frame*.
The above inequality can be understood as an "approximate Plancherel formula", characterizing the preservation of energy by the transform and leading to the invertibility of the frame operator S:

$$Sf = \sum_{k,m \in \mathbb{Z} \times \mathbb{Z}} \langle f, \varphi_{k,m} \rangle \varphi_{k,m} \quad (5)$$

Note that the frame operator is usually defined as an operator on $L^2(\mathbb{R}^d)$ and the relation to the finite discrete case is actually of interest in itself, see [3, 4] for more details. Since we only consider the finite disrete case, which is of interest for implementation, we may think of S as a matrix mapping $\mathbb{C}^L$ to $\mathbb{C}^L$.
The invertibility of S is equivalent to the existence of frame bounds $0 < A, B < \infty$ in the frame inequality in (4). The invertibility of S, now, leads to the existence of so-called dual frames, yielding convenient reconstruction formulas. This can easily be seen as follows: The canonical *dual frame* $\tilde{\varphi}_{k,m}$, is given by

$$\tilde{\varphi}_{k,m} = S^{-1} \varphi_{k,m}. \quad (6)$$

For Gabor frames, the elements of the dual frame $S^{-1}\varphi_{k,m}$ are generated from a single function (the dual window $\tilde{\varphi}$), and will hence be denoted by $(\tilde{\varphi}_{k,m})$. This follows from the fact that S and S^{-1} (the frame operator and its inverse) commute with the modulation and translation operators M_{mb} and T_{ka}, for $m = 1, \ldots, M$ and $k = 1, \ldots, K$, see e.g. [5]. Hence,

$$f = S^{-1}Sf = \sum \langle f, \varphi_{k,m} \rangle \tilde{\varphi}_{k,m}. \quad (7)$$

The coefficients used in (7) are called *canonical* in order to distinguish them from (infinitely many) other possible expansion coefficients with respect to the same frame. In the case of a tight frame, $S = AI$, where I denotes the identity operator, and therefore $S^{-1} = \frac{1}{A} I$. Tight frames will be further discussed in the next subsection.
In the finite discrete case of $f \in \mathbb{C}^L$ a collection $\{\varphi_{k,m}\} \subset \mathbb{C}^L$ with $N = KM$ can only be a frame, if $L \leq N$ and if the matrix G, defined as the $N \times L$ matrix having $\overline{\varphi_{k,m}}$ as its $(n + kM) - th$ row, has full rank. In this case, the frame bounds are the maximum and minimum eigenvalues of the frame operator $S = G^* \cdot G$. Here, G^* denotes the adjoint of G. The eigenvalues of this positive matrix yield information about numerical stability. The closer the frame-bounds are, the closer the frame operator will be be to a diagonal matrix. If A and B differ too much, the inversion of the frame operator is numerically unstable.
In applications in audio signal processing, redundancy of 2, 4 or even higher is common. Further, the effective length of the window φ equals or is shorter [1] than the FFT-length. In this special situation, the frame operator takes a surprisingly simple form:
From the definition of the frame operator

$$Sf = \sum_{k,m} \langle f, \varphi_{k,m} \rangle \varphi_{k,m}$$

a straight-forward calculation (see [3] for details) shows that the single entries of S are given by

$$S_{ji} = \begin{cases} M \sum_{k=0}^{K-1} T_{ka}\varphi[j]\overline{T_{ka}\varphi[i]} & \text{if } |j-i| \bmod M = 0 \\ 0 & \text{else} \end{cases} \quad (8)$$

Since $M \geq l$, where l is the window-length, $j = i$ is the only case for which $|j - i| \bmod M = 0$ holds and $\varphi[j]$ and $\varphi[i]$ are both non-zero. Therefore, the frame operator is diagonal and the dual window $\tilde{\varphi}$ is calculated as

$$\tilde{\varphi}[n] = \varphi[n] / (M \sum_{k=0}^{K-1} T_{ka} |\varphi[n]|^2)$$

2.1 Tight frames: synthesising with the analysis window

As mentioned above, for a *tight frame*, the frame operator equals identity up to a constant factor. This is as close as we may get to an orthonormal basis. As a matter of fact, for any given Gabor frame, a corresponding tight frame can be found and, as for dual frames, by a surprisingly simple formula in many situations of practical relevance.
Note that the frame operator S is a positive and symmetric and therefore selfadjoint operator, from which it follows

[1] E.g. in the case of zero padding.

that S^{-1} and $S^{-\frac{1}{2}}$ are selfadjoint as well and commute with time-frequency shifts.
These properties allow the following manipulations of expansion (7):

$$\sum_{k,m}\langle f,\varphi_{k,m}\rangle\tilde{\varphi}_{k,m} = S^{-1}Sf = S^{-\frac{1}{2}}SS^{-\frac{1}{2}}f$$
$$= \sum_{k,m}\langle f, S^{-\frac{1}{2}}\varphi_{k,m}\rangle S^{-\frac{1}{2}}\varphi_{k,m} = \sum_{k,m}\langle f,\varphi^t_{k,m}\rangle\varphi^t_{k,m}$$

Remark 1 *Note that the tight window given by $\varphi^t = S^{-\frac{1}{2}}\cdot\varphi$ is closest to the original window in the following sense: Let φ be a window generating a frame for lattice constants a and b and let φ^t be the tight window given as $\varphi^t = S^{-\frac{1}{2}}\varphi$. Then for any function h generating a tight frame for lattice constants a and b, the following holds [6]:*

$$\|\varphi - \varphi^t\|_2 \leq \|\varphi - h\|_2$$

This result shows that the tight window calculated as $\varphi^t = S^{-\frac{1}{2}}\varphi$ combines the advantage of using the same window for analysis and synthesis with optimal similarity to a given analysis window. At the same time no "correction" by multiplication with a gain function is necessary after processing, which makes processing more efficient and the results less ambiguous in the case of modification of the synthesis coefficients. This property becomes even more relevant, if the canonical Gabor coefficients are modified in some sense before resynthesis, e.g. in the case of time-frequency masking. In this case, the choice of a tight frame for analysis and synthesis minimizes the error arising from sampling in the coefficient domain. In the case of sparse coefficients, which we consider in the next section, tight frames also allow for a reliable interpretation of the obtained coefficients as well as satisfying reconstruction from these coefficients.

In analogy to the dual window and under the same conditions, we may deduce that the tight window φ^t corresponding to a given window φ and the time constant a can be calculated as:

$$\varphi^t[n] = (S^{-\frac{1}{2}}\varphi)[n] = \varphi[n]/\big(M\sqrt{\sum_{k=0}^{K-1} T_{ka}|\varphi[n]|^2}\big)$$

3. ENFORCING SPARSITY BY AN ℓ^1 CONSTRAINT

Being convinced that the signal components of interest have a sparse, at least approximative, representation in the atomic systems we use, we may directly look for relevant coefficients only. The prior information can be introduced by assuming an appropriate distribution of the coefficients. Mathematically, minimization of an ℓ^1-constraint on the coefficients yields explicit solutions and fast algorithms. [2]
In fact, the ℓ^1-constraint corresponds to a prior on the coefficients $\mathbf{c}$ of the form

$$p(\mathbf{c}) = \exp(-\mu\|\mathbf{c}\|_1).$$

[2] Note, that it has been proved that certain situations ℓ^1-minimization in fact yields the optimally sparse solution, see [7].

This prior leads to the following minimization problem. Given a tight Gabor frame with elements $\varphi^t_{k,m}$, we wish to minimize the following expression:

$$\Delta(x) = \|\sum_{k,m} c_{k,m}\varphi^t_{k,m} - \hat{x}\|_2^2 + \mu\|\mathbf{c}\|_{\ell^1} \qquad (9)$$

where $\|\mathbf{c}\|_{\ell^1} = \sum_{k,m}|c_{k,m}|$ is the ℓ^1-norm of the coefficient sequence and $\hat{x} = x + n$ is the observed signal, possibly contaminated by noise n. For orthonormal bases (instead of frames), the problem formulation in (9) leads to a well-known soft-thresholding solution. However, in the over-complete situation of frames, the situation is more intricate and an iterative procedure has to be applied.

3.1 Landweber iterations

While the classical basis pursuit [8] may be solved by linear programming algorithms, iterative thresholding is commonly used to solve the minimization problem posed in (9). Other algorithms exist, see [9, Chapter 12] for a thorough overview, however, the Landweber algorithm proposed in the current situation appeals by its simplicity and easy implementation. In the statistics community the iterative soft thresholding has been known for some time under the name "the lasso", [10]. The choice of μ is usually a delicate task and mirrors assumptions on the noise variance present in the signal model. In fact, increasing μ corresponds to the assumption of a higher noise variance and will thus lead to a sparser solution.
Let G_{φ^t} denote the Gabor analysis matrix corresponding to the tight system at hand and let $G^*_{\varphi^t}$ denote the corresponding synthesis system which is just the adjoint of G_{φ^t} in the case of a tight frame. To find the solution of (9), consider the sequence of iterates

$$\mathbf{c}^n = \mathbb{S}_\mu(\mathbf{c}^{n-1} + G_{\varphi^t}(\hat{x} - G^*_{\varphi^t}\mathbf{c}^{n-1})), \qquad (10)$$

where $\mathbf{c}^n$ are the expansion coefficients obtained in the n^{th} step, $\mathbf{c}^0$ is arbitrary and the thresholding operator $\mathbb{S}_\mu$ is given by

$$\mathbb{S}_\mu(c_{k,m}) = \begin{cases} c_{k,m} + \frac{\mu}{2}, & c_{k,m} \leq -\frac{\mu}{2} \\ 0 & |c_{k,m}| < \frac{\mu}{2} \\ c_{k,m} - \frac{\mu}{2}, & c_{k,m} \geq \frac{\mu}{2} \end{cases} \qquad (11)$$

It should be noted that $G_{\varphi^t}\hat{x}$ are the coefficients of the original data, which have to be calculated just once. The iterations thus consist of a gradient step (10), in which the coefficients are updated, and the soft thresholding step given in (11). Usually a stopping criterion is built into the algorithm using a fixed tolerance for $\|\mathbf{c}^n - \mathbf{c}^{n-1}\|$, unless a maximum number of iterations is reached before the stopping criterion is met.
According to [11], the corresponding iterative algorithm converges to the solution of (9).

3.2 Two examples

We first consider a synthetic signal comprised of two sinusoids with frequencies 1300Hz and 1400Hz, given a sampling rate of 8192. We use a Gaussian window of 400

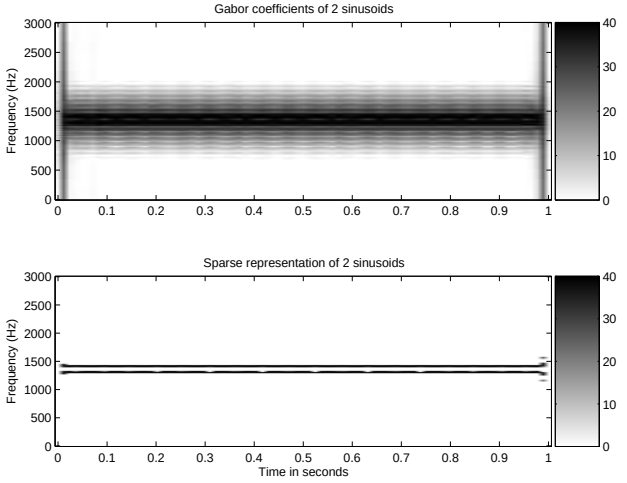


Figure 1. Gabor coefficients and sparse representation of two close sinusoids

samples length, the time-shift parameter $a = 100$ and calculate the canonical Gabor coefficients, shown in the first display of Figure 1. The second display, then, shows the coefficients resulting from ℓ^1-penalization on the expansion coefficients according to (9). It is immediately obvious, that the algorithm visually separates the two signal components. Note that approximate reconstruction from these coefficients is possible, as the correct phase factors are generated by the algorithm. A small error occurs, depending on the choice of μ in (9).

Our second example is a short extract from a music signal consisting of a piano, a double-bass and drums, see Figure 2. Again, we calculate the sparse coefficients, once with a wide Gaussian window, to represent the tonal parts, and once with a narrow Gaussian window to obtain sparser coefficients for transient parts. The results are shown in the 2nd and last display. Reconstructing from sparse coefficients, obtained with the wide window, yields a rather satisfying reproduction of the tonal part (bass, piano), while the residual coefficients mainly contain the cymbals' contribution. Note that, to this point, we have not applied any sophisticated statistical model to suppress noise or separate signal components, nor have we used any more sophisticated sparsity constrained as suggested in [12] to better encode dependencies between the single coefficients. We only use the fact that a relevant part of the signal has a sparse representation in the frame used for analysis. This example underlines the possible merits of the approach to the task of separating components in the time-frequency domain. Similar results using other overcomplete dictionaries have been obtained before, see e.g. [13], where sparse representations of signals were considered using the modulated complex lapped transform (MCLT). Important issues concerning the efficient encoding of the positions and values of the significant coefficients that arise in obtaining sparse coefficients such as those mentioned in e.g. [13, 14] will not be discussed in this paper. Let us just remark, however, that the number of significant coefficients in our experiments amounts to 0.3% to 3% of the size of

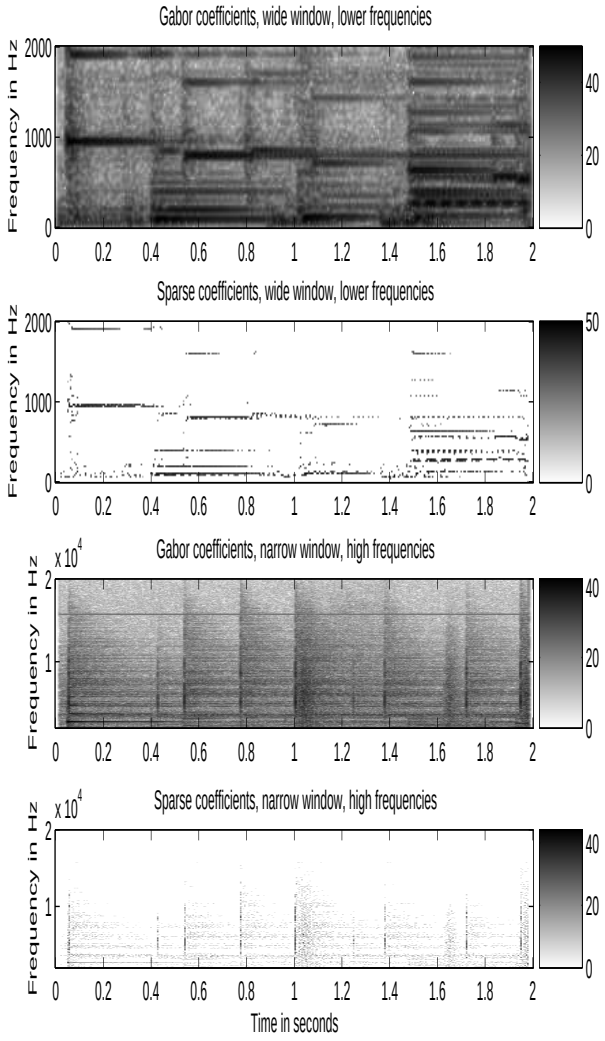


Figure 2. Sparse coefficients for a music signal

the coefficients space.

4. APPLICATION: AUDITORY SCENE DESCRIPTION

Our next application is in the area of auditory scene description, i.e. the classification of audio sources in sound mixtures of several sources. Please note that we are not aiming at source separation but at the easier task of source identification. On the other hand we are going beyond recognition of instrumental sources [15] by including a wider variety of sounds [1]. Also, although closely related to the work done in the very active research domain of music transcription, see, e.g. [16, 17], our approach addresses a slightly different task. Since our primary interest is in electro-acoustic music, for which often well defined sound grains are either pre-defined or can be extracted from a given composition, we wish to automatically determine, whether a particular source sound is present at a specific time in the piece. This is an important step in the process of automatic or semi-automatic annotation of electro-acoustic

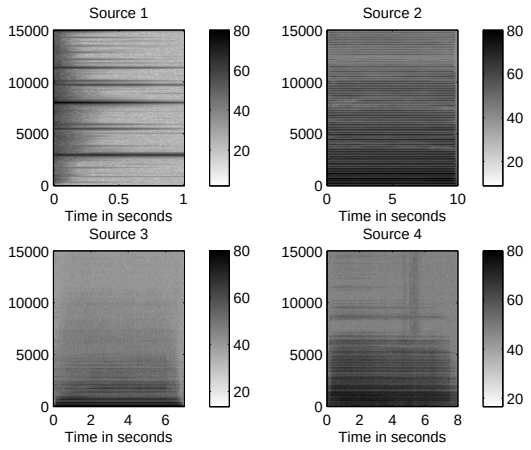


Figure 3. Canonical coefficients of four sources

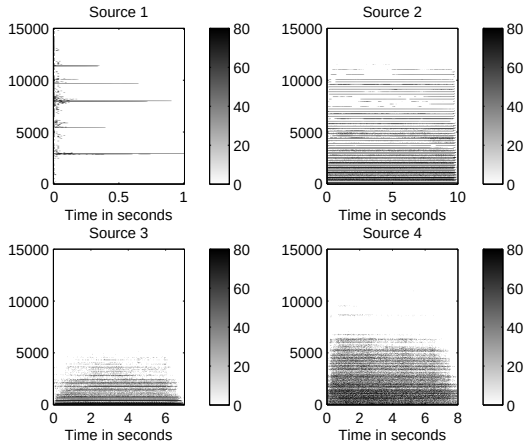


Figure 4. Sparse coefficients of four sources

music.

4.1 Sound Sources

The sources in our example are:

- source 1: single high note played on a Glockenspiel (one second long)
- source 2: long note played on an accordion of medium pitch (ten seconds long)
- source 3: long clarinet tone, low pitch (seven seconds long)
- source 4: noisy clarinet sound, without pitch, air only (eight seconds long)

The canonical and sparse coefficients of the sources are shown in Figure 3 and Figure 4, respectively.

While sources 2 and 3 are harmonic sounds produced by instruments, source 1 is an inharmonic sound produced by a bell and sound 4 is a noisy sound with little harmonic constituents. These sources were chosen to reflect the wide variety in spectral and temporal characteristics displayed by sounds commonly used in electro-acoustic composition.

4.2 Method

We consider the following setting: we are given a sound file generated by N known sources s^j, $j = 1, \ldots, N$, which can be active for any given time t. The signal then is a sum of shifted copies of the sources s^j at time t_k, i.e. $s^j(t-t_k)$. For the purpose of our experiment however, we approximate f at uniform overlapping time intervals I_l as a linear combination of the sources:

$$f_{I_l} \approx \sum_{j=1}^{N} a_{j,I_l} s^j, \tag{12}$$

where a_{j,I_l} is a function storing the activation pattern, i.e. a binary function with values in $\{0, 1\}$. We wish to recover a_{j,I_l} for $j = 1, \ldots, N$, over the intervals I_l.
To do so, we observe the following. Since we expect approximate orthogonality of the various sources in the transform domain, we may assume that the correlation between the coefficients of the mixture and each of the sources reflects the presence of the sources. We therefore proceed as follows: time-frequency coefficients of both the source specimen and the mixture are being computed; overlapping time slices of the time-frequency coefficients are correlated with time-slices of the same length from all four source specimen.

In the sequel we are going to use the absolute values of both the canonical Gabor coefficients and the sparse coefficients $c_{k,m}$ obtained as solution of (9). For brevity, we set $\hat{s}_{k,m} = |\mathcal{V}_\varphi s(k,m)| = |\langle s, \varphi_{k,m}\rangle|$ and $\hat{c}_{k,m} = |c_{k,m}|$. In order to judge the approximate orthogonality of the source specimen in the coefficient domain, we define the inner product of coefficient matrices $\hat{s}^1, \hat{s}^2$ as

$$\langle \hat{s}^1, \hat{s}^2 \rangle_M = \sum_k \sum_m \hat{s}^1_{k,m} \cdot \hat{s}^2_{k,m}$$

and consider the following matrices:

$$CM_{i,j} = \langle \hat{s}^i, \hat{s}^j \rangle_M, \text{ and } CM^{spars}_{i,j} = \langle \hat{c}^i, \hat{c}^j \rangle_M.$$

We normalize the coefficients corresponding to the various sources, so that we can say that deviation from orthogonality between the sources is reflected in deviation from diagonality of the matrices CM and CM^{spars}, respectively. On the other hand, if the condition number of the obtained matrices is good, the correlation between sources can be corrected by applying the inverse of the respective matrix to the obtained correlations between mixture and sources. For clarity, we describe the de-correlation step for time-frequency coefficients $\hat{f}$ of any signal f without specifying whether the coefficients are canonical or sparse for the moment. For the mixture signal f we observe:

$$\langle \hat{f}_{I_l}, \hat{s}^k \rangle \approx \sum_{j=1}^{N} a_{j,I_l} \langle \hat{s}^j, \hat{s}^k \rangle,$$

so that in order to retrieve the coefficients a_{j,I_l}, we have to solve a system of equations involving the matrices CM or CM^{spars}. The detailed procedure is described in Section 4.3 below.

- Note that it is vital in our method to consider time-frequency coefficients rather than just single spectral representations. This takes the time-structure of the signals into account. This procedure makes it unnecessary to examine the correlation coefficients in every time-instant.
- Since the time-structure of the sources is essential to the performance of our method, transient signal components, in particular clicks, are not well-described. However, methods for extraction of transient signal components exist, see e.g. [18, 19, 20, 21], and therefore, their classification may be considered separately.
- For the simulations discussed below, we chose time-slices of one second length and an overlap of 0.5 seconds. As we will see, for sources with significant time-structure, this approach yields rather satisfying results.

4.3 Experiment and results

In the experiments, we consider a one minute signal mix consisting of the four sources mentioned above, at most three of which are active at any time. We calculate the Gabor coefficients of the whole length of the mix (one minute) but just one second for each of the sources using the following parameters: a Hanning window of length 1024 samples (corresponding to $23ms$ at a sampling rate of $44100Hz$) with a hop size of 512, from which a tight window is obtained. This yields a Gabor coefficient matrix of size 1024×5169 for the signal mix and 1024×88 for each of the sources. We then consider time-slices $\hat{f}_{I_l}$ of the Gabor coefficient matrix of the same size as the coefficient matrix of the sources, hence corresponding to a duration of 1 second. We consider an overlap factor of 2 between subsequent time-slices, resulting in 116 time positions in our setting. We then compute the correlation-matrix C, of size 6×116, whose entries are given by

$$C_{j,l} = \langle \hat{f}_{I_l}, \hat{s}^j \rangle_M .$$

Since the model for each time-slice is approximated to be a linear combination of the sources, solving for the coefficients a_{j,I_l} amounts to computing for $\mathrm{inv}(CM) * C$. For the four sources in our experiments, we obtain condition numbers 2.9936 and 1.6425 for CM and CM^{spars}, respectively, which reflects, in this case, their deviation from the identity. This can be interpreted by saying that the sparse coefficients of the sources have less overlap than the canonical coefficients, as expected.
The resulting activation matrix is then normalized for each source, such that the maximum value assumed is 1 for each source. Hence the same threshold is simultaneously applied to all sources, entries above the threshold are set to 1 while the rest are set to 0. The same procedure is applied to the sparse representation of the signal and the sources. The sparse coefficients have been obtained by applying Landweber iterations with $\mu = 0.04$.
Figure 5 shows the true map of time positions where each source is present (target map, black indicating presence of a source), as well as the matrices obtained from the above mentioned procedure, using the Gabor coefficients and the sparse representation with threshold values 0.145 and 0.2 (with the lower threshold being optimal for the sparse representation and the higher for Gabor coefficients).

In comparing the resulting matrices with the true maps, we analyze the receiver operating characteristic (ROC) curve and compute the following:

- accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- specificity $= \frac{TN}{TN+FP}$
- sensitivity $= \frac{TP}{TP+FN}$,

where TP, TN, FP, FN signify true positives, true negatives, false positives, and false negatives, respectively. We see in Figure 6 the corresponding graphs plotted over varying threshold values.

An ideal method for identifying the sources in the mixture would have both specificity and sensitivity equal to one. In more realistic settings it is necessary to find a threshold where a good compromise between high specificity and sensitivity exits. As can be seen from the rightmost plot in Figure 6, both the canonical and sparse representations yield almost equal results in terms of sensitivity. With increasing threshold, less and less sources are detected correctly. But as can be seen from the middle plot in Figure 6, the optimum value for specificity is reached earlier for the sparse representation. This means that one can choose a threshold where specificity is optimal (i.e. no false positives) while still having very high sensitivity (i.e. high amount of true positives). This also results in the optimum value in terms of accuracy being reached earlier for sparse representations (leftmost plot in Figure 6). These slightly improved results are due to the lower cross-correlation between signal components in the sparse representation.

5. CONCLUSIONS AND PERSPECTIVES

We suggested to apply a sparsity-promoting norm on the coefficients in a Gabor expansion. We also recalled how to calculate dual and tight Gabor frames for the situation most commonly encountered in audio signal processing. It seems highly recommendable to prefer tight frames if modification of the canonical coefficients is envisaged. In an application to classification of sound sources, experimental results indicate that sparse coefficients help to avoid false positives as compared to the results obtained from using canonical Gabor coefficients. Since sensitivity is comparable for both sets of coefficients, choosing a sparse representations leads to slightly better over-all classification results. The influence of various parameters, in particular the effects of the threshold in the Landweber iterations is yet to be investigated. Future work will also

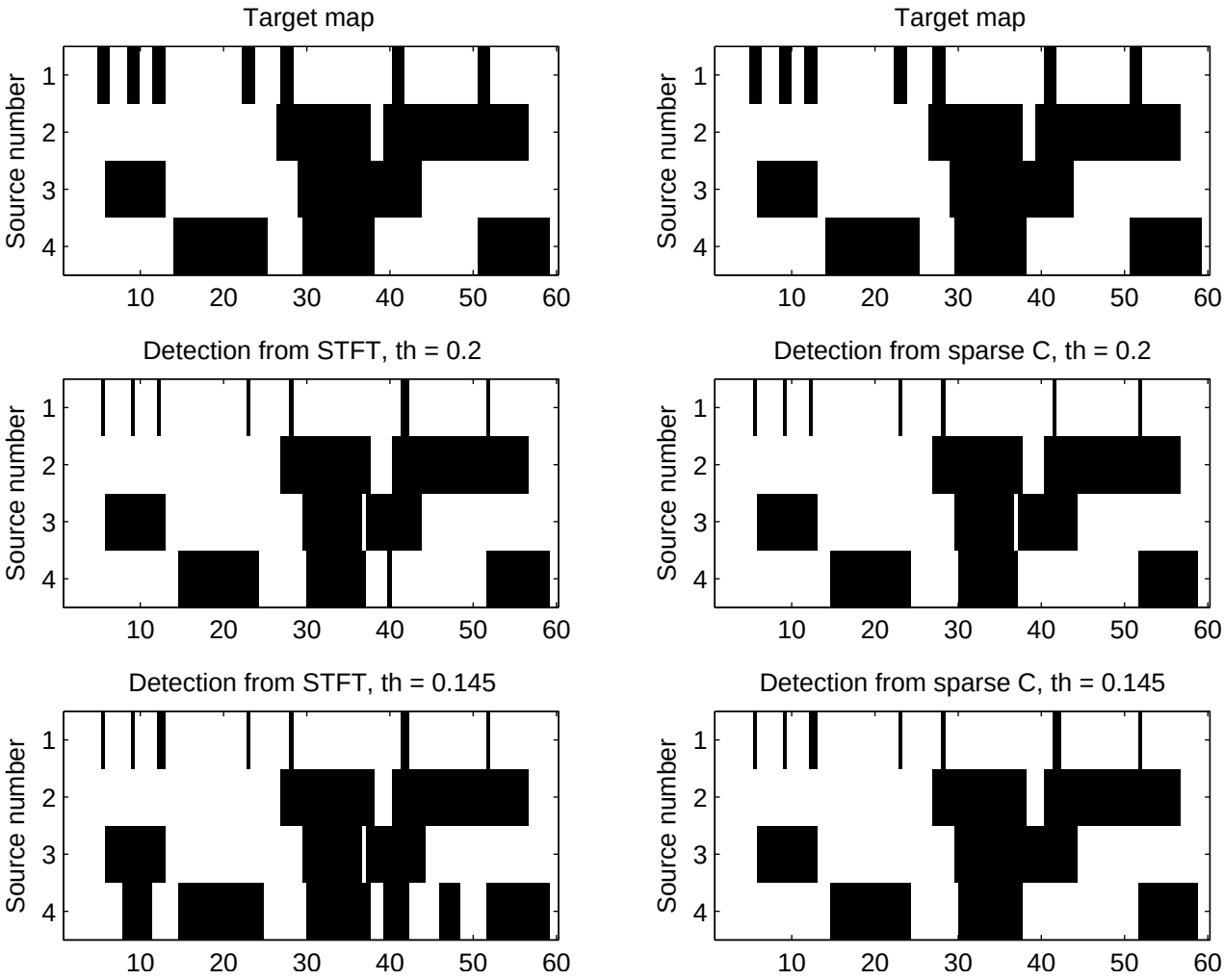


Figure 5. Detection from canonical and sparse coefficients

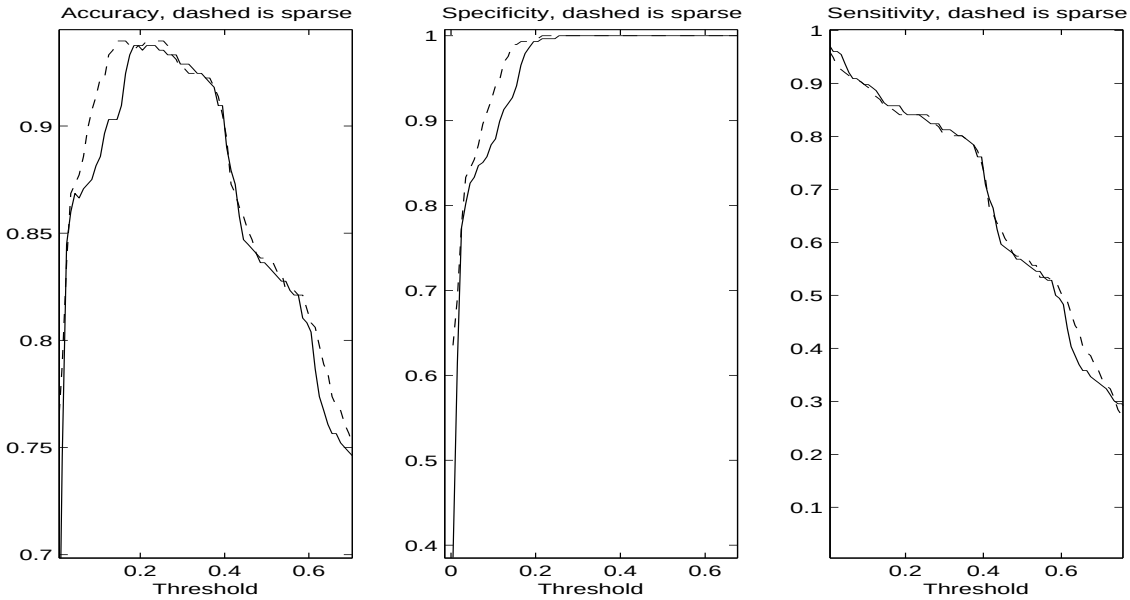


Figure 6. Evaluation of detection from canonical and sparse coefficients

include the application of more sophisticated coefficient norms as suggested in [12] as well as the usage of frames other than Gabor frames, e.g. wavelets, in order to include transient components. Furthermore, systematic evaluation on a more extensive data base will allow us to judge the influence of the various parameters involved.

6. ACKNOWLEDGMENTS

This research is supported by the Austrian Science Fund (Projects T384-N13 and P21247) and the Vienna Science and Technology Fund (WWTF, project Audiominer).
We thank the anonymous reviewers for their valuable comments and suggestions.

7. REFERENCES

[1] D. Hoiem, Y. Ke, and R. Sukthankar, "Solar: Sound object localization and retrieval in complex audio environments," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, pp. 429 – 432, March 2005.

[2] H. G. Feichtinger and T. Strohmer, *Gabor Analysis and Algorithms. Theory and Applications.* Birkhäuser, 1998.

[3] M. Dörfler, "Time-frequency Analysis for Music Signals. A Mathematical Approach," *Journal of New Music Research*, vol. 30, no. 1, pp. 3–12, 2001.

[4] N. Kaiblinger, "Approximation of the Fourier transform and the dual Gabor window," *J. Fourier Anal. Appl.*, vol. 11, no. 1, pp. 25–42, 2005.

[5] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis.," *IEEE Trans. Inform. Theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[6] A. J. E. M. Janssen and T. Strohmer, "Characterization and computation of canonical tight windows for Gabor frames.," *J. Fourier Anal. Appl.*, vol. 8, no. 1, pp. 1–28, 2002.

[7] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l^1 solution is also the sparsest solution," *Commun. Pure Appl. Anal.*, vol. 59, no. 6, pp. 797–829, 2006.

[8] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by Basis Pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

[9] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, 2009.

[10] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[11] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[12] M. Kowalski and B. Torrésani, "Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients," *Signal, Image and Video Processing, doi:10.1007/s11760-008-0076-1*, 2009.

[13] M. Davies and L. Daudet, "Sparse audio representations using the MCLT," *Signal Process.*, vol. 86, pp. 457–470, March 2006.

[14] E. Ravelli, G. Richard, and L. Daudet, "Union of MDCT bases for audio coding," *IEEE Trans. Audio Speech Lang. Process.*, vol. 16, no. 8, pp. 1361–1372, 2008.

[15] M. Goto, "Music scene description," in *Signal Processing Methods for Music Transcription* (A. Klapuri and M. Davy, eds.), pp. 327–359, New York: Springer, 2006.

[16] A. Klapuri and M. Davy, *Signal Processing Methods for Music Transcription*. Springer, 2006.

[17] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 177–180, IEEE, 2003.

[18] F. Jaillet and B. Torresani, "Timefrequency jigsaw puzzle: adaptive multiwindow and multilayered Gabor expansions," *Int. J. Wavelets Multiresolut. Inf. Process.*, vol. 2, pp. 293–316, 2007.

[19] S. Molla and B. Torrésani, "A hybrid scheme for encoding audio signal using hidden Markov models of waveforms.," *Appl. Comput. Harmon. Anal.*, vol. 18, no. 2, pp. 137–166, 2005.

[20] J. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Processing Letters*, vol. 11(6), pp. 553–556, June 2004.

[21] B. Torrésani and S. Molla, "Transient Detection and Encoding Using Wavelet Coefficient Trees,," in *proceedings of the GRETSI'01 conference* (F. Flandrin, ed.), 2001.

MUSICJSON: A REPRESENTATION FOR THE COMPUTER MUSIC CLOUD

Jesus L. Alvaro

Computer Music Lab
fauno.org, Madrid, Spain

JesusLAlvaro@gmail.com

Beatriz Barros

Dpto. Lenguajes y Ciencias de la Computación
UMA, Málaga, Spain

bbarros@lcc.uma.es

ABSTRACT

New cloud computing ways open a new paradigm for music composition. Our music composing system is now distributed on the Web shaping what we call as *Computer Music Cloud* (CMC). This approach benefits from the technological advantages involved in distributed computing and the possibility of implementing specialized and independent music services which may in turn be part of multiple CMCs. The music representation used in a CMC plays a key role in successful integration. This paper analyses the requirements for efficient music representation for CMC composition: high music representativity, database storage, and textual form. Finally, it focuses on its textual shape, presenting *MusicJSON*, a format for music information interchange among the different services composing a CMC. MusicJSON and database-shaped representation, both based on an experienced sound and complete music representation, offer an innovative proposal for music cloud representation.

1 . INTRODUCTION

Cloud Computing, a new term defined in varied ways [7], involves a new paradigm in which computer infrastructure and software are provided as a service [5]. This services themselves are referred to as *Software as a Service (SaaS)*. Google Apps is a clear example of *SaaS* [8]. Computation infrastructure is also offered as a service (*IaaS*), thus enabling the user to run the customer software.

This new paradigm offers new possibilities for the design of composition systems. Fig. 1 shows the Computer Music Cloud (CMC) approach where the system is distributed across specialized online services [2]. The user interface is now a web application running in a standard browser (1). A storage service is used as an edition memory (2). An intelligent-dedicated service is allocated for music calculation and development (3). Output formats such as MIDI, graphic score and sound file are rendered by independent services exclusively devoted to this task (4). The web application includes user sessions to allow multiple users to use the system. Both public and user libraries (5) are also provided for music objects. Intermediary music elements can be stored in the library and also serialized into a *MusicJSON* format file, as described below.

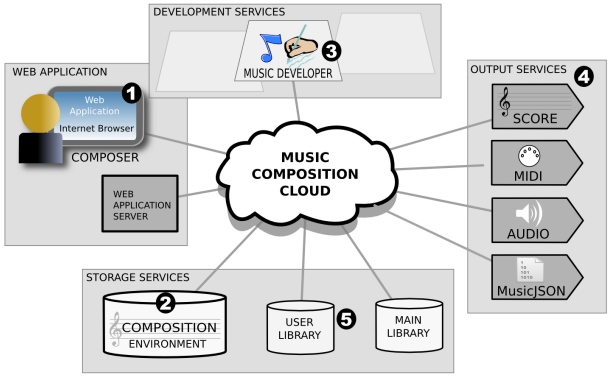


Fig. 1. Basic Structure of a Composition Music Cloud

This CMC approach has several advantages. Some of them come from Cloud Computing, such as scalability, optimization and reuse of available resources. Others come from web applications, such as decentralized information, being able to work from any computer with a standard Internet connection and browser, and the inherited code. Also, the division of the music system into services allows for the design and implementation of independent services with the most appropriate tools, It is apart from the availability of a service for different CMC systems. Since services can be shared, the design of new music systems is facilitated by the joint work of different services controlled by a web application.

The key factor in successful integration is the use of a well-defined music representation for music data interchange. This is also the objective of this paper: presenting a proposal for the interchange of music information among the services which shape a music composition cloud. In the next section, the types of services included in this cloud are analysed as a base to define the requirements to be fulfilled by the selected music representation, tackled in Section 3. Section 4 describes the MusicJSON format as a textual form of the used representation, while Section 5 describes some use examples. The paper ends with some conclusions and some points referred to related work.

2 . MUSIC SERVICES IN THE CMC

In a simple form, Music Web Services are servers receiving a request and performing a task. At the end of a task the resulting objects are returned to the stream

or stored in an interchange database. The access to this database is a valuable feature for services since it is a shared workspace where the componets of music composition are represented. The Music Services of the cloud can be classified according to their function. These services are described in the following subsections.

2.1 Input

This group includes the services aimed particularly at incorporating new music elements and translating them from other input formats.

2.2 Agents

They are those services which are capable of inspecting and modifying music composition, as well as introducing new elements. They include human user interfaces, but they may also consider other intelligent elements taking part in music composition[4]: introducing decisions, suggestions or modifications. In our prototype, we have developed a web application [2] acting as a user interface through the edition of music objects.

2.3 Storage

There are two main types of storage services: libraries and composition environments. Libraries store music objects which shall be used in different compositions. There are general libraries and user libraries. *Main lib* stores shared music elements as global definitions. This content comprises music elements shared by all users as a shared music language. User-related music objects are stored in the *User lib* and include composer-defined music objects which can be reused in several parts or compositions.

Composition environments are the storage services where the piece is progressively composed. This database contains the composition environment (i.e., everything related to the piece currently under composition), and does not only act as a space for information interchange, but also as a real and shared music environment with which several services can interact simultaneously and coordinatedly.

2.4 Development

The services in this group perform *development* processes. As explained in [1], *development* is the process by which higher-abstraction symbolic elements are turned into lower-abstraction ones. High-abstraction symbols are implemented as *meta-events* and represent music objects such as *motives*, *segments,* and other composing abstractions [1]. Algorithmic composition developers and other intelligent services, such as constraint solvers or genetic algorithms, are examples of this type of music development service .

2.5 Output

These services produce output formats as a response to requests from other services. They render formats from the element currently under edition for immediate composer feedback as well as the whole score or audio. The MIDI file for audio playing and standard notation in a graphic format are two clear examples of this. Other output formats are also possible by integrating a suitable translation service.

3 . MUSIC REPRESENTATION FOR THE CMC

To achieve an effective integration of all elements in the cloud for music composition, all services must share the same music representation. This representation must meet three main requirements: satisfactorily represent the basic elements of music composition in a solid hierarchy of classes; present a representation form for storing music objects in a database; and count on a textual form which facilitates the interchange of music objects among the different services. Besides, the cloud's distributed nature must also be taken into account by incorporating the possibility of distribution in the data. All these elements are described in the following subsections.

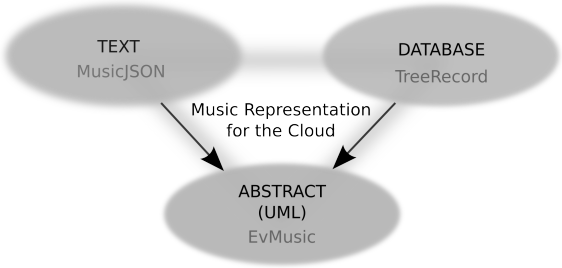


Fig. 2. Abstract, Database and Textual Forms of Music Representation

3.1 EVMusic Representation

The proposal is based on EVMusic representation [1]. It is a robust model with high multi-level representativity, multiple topologies and meticulous, detailed representation of music pitch, as well as full compatibility with traditional music notation. Likewise, its multi-level nature and expandability allow for representing music elements at varied abstraction levels.

EVMusic representation counts on a complete hierarchical organization of classes designed in the platform-independent UML [13], which allows for its use in multiple programming languages. Figure 3 shows a brief extract from the UML representation, showing only some of the classes present in the music fragment in Figure 6, which shall work as a reference for subsequent examples. The present paper is not aimed at contributing a detailed analysis of EVMusic classes. Nevertheless, we shall contribute a brief description of some of the most relevant aspects shown in this figure, particularly the relations.

In a UML Class Diagram [12], the inheritance relation among classes is represented with a hollow pointed arrow. Thus, it can be observed in this figure that a *scoreelement* is an *event*, which in turn is a *treeobj* just like *singlepitch*. Inheritance relations can also be multiple, as it occurs with

the note class. Thus, it can be deduced that a note is either a score element with a unique pitch ,or a pitch temporarily placed on the score.

Containment relations are indicated by means of rhombus-shaped arrows and the name of the relation. Thus, it can be observed in the figure that an object of the *aggregate* class includes an attribute known as *pitches*, which is a container of *singlepitch*-like elements. Therefore, a chord (an *nchord*-like object) which inherits the *aggregate* properties is a *scoreelement* which contains several *singlepitches*. Grace notes are also represented with a content relation: they are a sequence of *singlepitches* which ornament a *scoreelement* and are stored in its slot known as *graces*. Finally, the structure of similar objects is also indicated by the content relation. In EVMusic, each event is by default a group of subevents, as clearly reflected in the fractal-like structure of music time. Thus, these subevents are stored in the slot *events* of the main event. The tree structure is represented by means of the *treeobj* class and its *parent* relation. It is extended to all representation elements and shall be commented on in the following section.

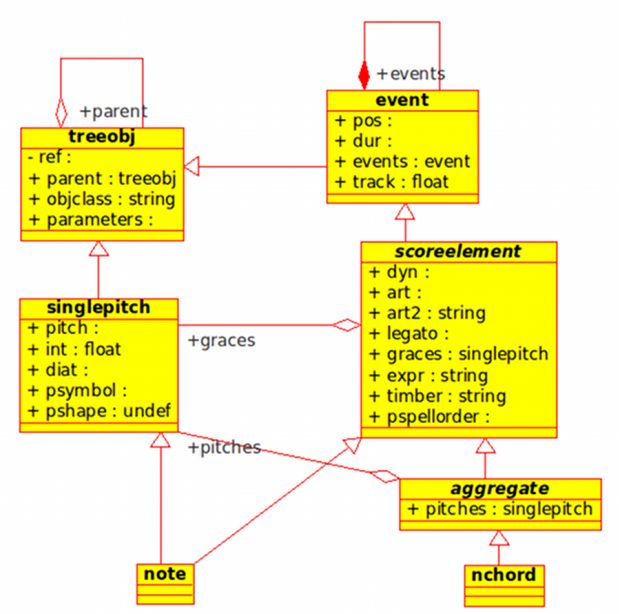


Fig. 3. UML Diagram with some classes of EvMusic Representation

3.2 Database Stored Representation

Database storage allows several services to share the same data and collaborate in the composition process. The information stored in a database is organized in tables of records. The database representation must achieve two objectives: store music instances and store the relations among these forms.

Figure 4 shows a simplified scheme with some database fields. To represent the objects of a particular class, the table of the database incorporates the field *objclass*, which contains the name of the object class. The attributes of each class are usually represented by fields with the name of the attribute. In this figure, these attributes are indicated in the rows of predefined attributes (*AttributeA, AttributeB,...*) Importantly, appropriate representations must enable the incorporation of new elements, even those which have not been defined yet. Some pairs of generic fields have been provided with this purpose. These pairs of generic fields store both the name of the attribute and its value. They are indicated in the central rows of this figure as Expandable Attributes. For instance, if we want to incorporate a new attribute known as "zattrib", the register will contain "zattrib" in the field *attrib1name,* and its value in the field *attrib1value*. Thus, implementing a new storage system is not necessary when a new class, with its new attributes, is defined.

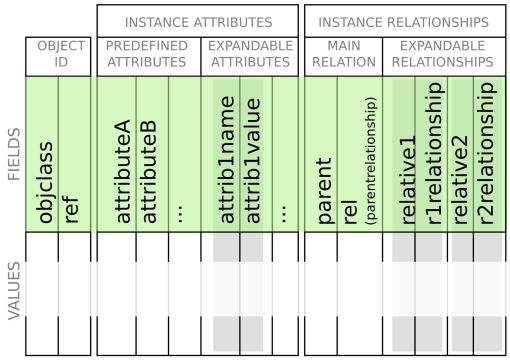


Fig. 4. Record Structure of Data Base

The relations among the music objects are also represented in the database. Among all of them, the *containment* or belonging relation is likely to be the most important one. As we have already mentioned, the music objects of the abstract EVMusic representation are usually tree-shaped related. To be stored in a database, these tree structures must be previously converted into records. For this purpose, the three main classes of EV representation are subclassed from a tree node class *treeobj*, shown in Fig. 3. Thus, every object is identified by a unique *reference* and a *parent* attribute. This allows to represent a large tree structure of nested events as a set of records for individual retrieval or update. By default, the slot *parent* always refers to the *containment* relation. However, other relation can be used for this main tree. The field *parentrelationship* (abbreviated as *rel*) was incorporated with this aim. For instance, the last row in Table 1, shows how the grace note indicates the main note *nt03* as *parent*, but the parent relationship in this case is not a temporary structure, but a grace-note structure, so its value in the field *rel* is "*grace*".

The representation of relations has been completed by incorporating new relatives. As it can be observed in Figure 4, the main relation *parent* was added pairs of fields aimed at indicating new relatives. Thus, for instance, a new relation r1 can be incorporated by indicating the reference of the referred object in the field *relative1* and the relation between them in the field *r1relationship*. The incorporation of new relations in the

database contributes an important degree of representativity since it opens new creative and representative possibilities, such as, for instance, the opportunity of representing *constraints* among music objects or the definition of some objects according to others. Multiple relations also allow for the coexistence of several organizations of objects, letting , for instance, the same note belong to both a temporary structure (represented by the relation *parent*) and a harmonic structure (represented by an *extended relation*) simultaneously.

ref	parent	objclass	pos	dur	track	pitch	pspellorder	legato	art	dyn	name	rel
sco01		score	0								Example	
sec01	sco01	section	1									
stf01	sec01	staff					1				Violin	
prt01	stf01	part	1		1							
	prt01	note	0	0,5		69			st	mf		
	prt01	note	0,5	0,5		69			st			
nt03	prt01	note	1	0,75		74		start				
	prt01	note	1,75	0,25		73		end				
	prt01	note	2	0,5		74			st			
	prt01	note	2,5	0,5		76			st			
nch08	prt01	nchord	3	1								
	nch08	spitch				78						
	nch08	spitch				69						
	nt03	spitch				76						grace

Table 1. Database Content for a simple example

Table 1 shows the database content for the music example notated in Figure 6. The objclass field indicates the class of every instance. Note the relation with the music example and the following listing code in Section 4.1

The music objects described in the database of a music storage service can belong either to the environment of the music piece currently under composition, or to a general or user library. Library objects are referred to by other objects by putting the prefixes *x.lib* and the reference of the library before the reference of the object, as we shall see in the following examples.

3.3 MusicJSON Textual Representation

The third specification which must be met by the CMC representation is counting on a textual form which allows music information to be interchanged among services through web streams.

When it comes to design an appropriate textual format for data, basing on formats already widely-used in the Internet seems a rather convenient strategy. Web applications usually use XML and JSON (Java Script Object Notation) [11] for data interchange. Both formats meet the requirements. XML has been successfully used for score representation [15]. However, we opted for JSON, mainly due to the large JSON-compatible tool library available at the time of writing this paper, and the fact that JSON is the interchange format for some of the main Internet web services such as Google or Yahoo. In addition, JSON provides great features such as human readability and dynamic unclosed object support, a very valuable feature inherited from the prototype-based nature of JavaScript [10]. To facilitate communication, JSON also offers JSONP [9] and its corresponding libraries for web applications, which extends interaction flexibility among web services.

As mentioned in [11], *"JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.[...] JSON is built on two structures:*

- *A collection of name/value pairs. In various languages, this is realized as an object, record, structure, dictionary, hash table, keyed list, or associative array*
- *An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence."*

These universal data structures in JSON can be used to describe EvMusic objects and communicate among web music services. *MusicJSON* is the name given to this use. Once the database-shaped representation has been detailed, MusicJSON is easily understandable since they are directly and closely related. MusicJSON can be understood as a serialization of database content. MusicJSON objects are therefore collections of key/value pairs which have been assigned the afore-described attribute *objclass*, so each object always declares its class.

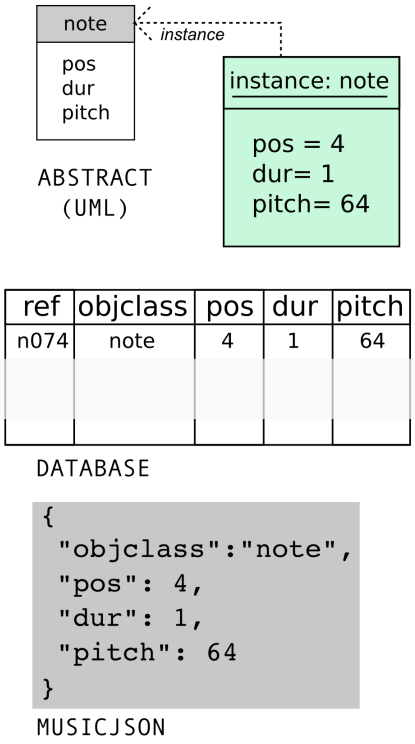


Fig. 5. A note instance in UML, database and MusicJSON representation

To give a simple example, Figure 5 shows the same object as an instance in a ULM diagram, as an entry in the database, and as a MusicJSON text. Please observe that the textual representation includes the attribute *objclass* with the value "note".

The code in Section 4.1 shows a more illustrative example applied to a brief music fragment. Compared to its standard notation in Figure 6, and Table 1, MusicJSON code is easily understandable. In order to improve the readability, some data semantics are allowed in MusicJSON, like the use of pitch names instead of numbers, as shown in the code.

In MusicJSON not all the attributes of the object are necessary, but only the relevant ones (i.e., those necessary for object definition and abstract-instance construction). If the values corresponding to a necessary property are not indicated, the default values of its corresponding class, or even specific default values defined for a particular group of objects shall be taken. The key *defaultcontent* is provided with this purpose.

Regarding structures, represented in the database as content relations among entries, MusicJSON is not a mere serialization of each database register. It can represent arrays or lists, so structures are presented directly in a deployed form. This tree structure can be observed in the listing code in Section 4.1, which spreads completely the tree *score -> section -> staff -> part -> note*.

Type	Description	MusicJSON use	Previous definition	Extended Reference

External	Use of an external object by its MusicJSON url	{
		"objclass":"xref",
		"type":"url",
		"url":
		"http://myobjects.com
		/mymotivc",
		"subclass":"motive"
		}
				x.url.http://myobjects.com
				/mymotive

Library	Use of an Object from a Library	{
		"objclass":"xref",
		"type":"lib",
		"lib": "main",
		"name": "mota",
		"subclass":"motive"
		}
			"lib": {
			"main":"http://mylibs.com
			/mainlib",
			...
			}
				x.lib.main.mota

Variable	Object defined into a variable	{
		"objclass":"xref",
		"type":"def",
		"name": "ma",
		"subclass":"motive"
		}
			"def":{
			"ma":
			{ "objclass":"motive",
			"symbol": [0,7,5,4],
			"slength": "+-+- +--+",
			},
			...
			}
				x.def.ma

Context	Use of a context object	{
		"objclass":"xref",
		"type":"ctx.mtime",
		"name": "scale",
		"subclass":"rpcs"
		}
				x.ctx.mtime.scale

Table 2. Extended references in MusicJSON

Other valuable feature of MusicJSON is related to its distributed nature: *extended references*. Like a hyper-document, MusicJSON allows for the use of external objects either from a library or directly. A new kind of *xref* object was defined with this purpose. The following table shows some types of *extended references* such as external objects by means of URL, elements of an imported library, elements defined in a variable, or contextual elements.

Referred objects can be used in two ways: either defining an xref-type object (as shown by the third column), or using directly the extended reference with the notation of prefixes separated by dots (as shown in the last column). The former section has already shown how library objects are referred to with the prefix *x-lib*. The extended reference for an external object is denoted by the prefix *x.url* followed by the URL of the object. The individual and direct references to a previously-defined object are indicated by the prefix *x.def* followed by the reference previously defined within the same context.

The incorporation of xref objects and *extended references* is an important value added to textual representation since it allows for combining varied elements which can be distributed in the Web. Arguably, this makes it a valuable feature for a knowlegde representation for the Cloud. In addition the use of *extended references*, is open to the definition of new references by using new prefixes. For instance, the last row in Table 2 shows a reference to the context of musical time, denoted by the prefix *x.ctx.mtime*. The time context of a music object is very important, as in harmony. For instance, an extended reference of this type enables the definition of an object which depends on the current harmonic context.

4 . MUSICJSON IN EXAMPLES

This section shows some illustrative examples of the use of MusicJSON.

4.1 Music Fragment in Traditional Notation

As previously stated, MusicJSON is compatible with the traditional notation. For the sake of illustration, we include a simple example of traditional notation of a brief music fragment and its corresponding representation in the MusicJSON format. All music objects in the example are represented in the UML diagram of Figure 3. Note MusicJSON's high readability and clear relation to traditional notation. Also note the generic tree structure of a score: *score -> section -> staff -> part -> note*, at the beginning of the code, and the relation to the database content in Table 1 representing the same music example.

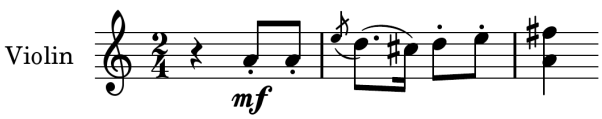


Fig. 6. Score notation of the example

```
{
  "objclass": "score",
  "pos": 0,
  "events": [{
    "objclass": "section",
    "pos": 1,
    "events": [{
      "objclass": "staff",
      "name": "Violin",
      "pspellorder": 1,
```

```
      "events": [{
        "objclass": "part",
        "track": 1,
        "pos": 1,
        "events": [{
          "objclass": "note",
          "pos": 0,
          "dur": "0.5",
          "pitch": 69,
          "art": "st"
          "dyn": "mf",
        },
        {
          "objclass": "note",
          "pos": 0.5,
          "dur": "0.5",
          "pitch": 69,
          "art": "st"
        },
        {
          "objclass": "note",
          "pos": 1,
          "dur": "0.75",
          "pitch": "d5",
          "legato": "start",
          "graces": [{
            "objclass": "spitch",
            "pitch": 76
          }]
        },
        {
          "objclass": "note",
          "pos": 1.75,
          "dur": "0.25",
          "pitch": 73,
          "legato": "end"
        },
        {
          "objclass": "note",
          "pos": 2,
          "dur": "0.5",
          "pitch": 74,
          "art": "st"
        },
        {
          "objclass": "note",
          "pos": 2.5,
          "dur": "0.5",
          "pitch": 76,
          "art": "st"
        },
        {
          "objclass": "nchord",
          "pos": 3,
          "dur": "1",
          "pitches": [{
            "objclass": "spitch",
            "pitch": "f#5"
          },
          {
            "objclass": "spitch",
            "pitch": 69
          }
          ]
        }
        ]
      }]
    }]
  }]
}
```

4.2 Storage Service Access

The interchange of musical information between the services of the CMC can be done directly, but it is also possible to exchange it in a shared form through the database. Any exchange of information with the Storage Service is done in MuiscJSON format; not just musical objects, but also the communication protocol. The storage service responds to standard GET requests with a URL that ends with the function to perform, usually a CRUD function (Create, Retrieve, Update, Delete). The request is accompanied by two parameters: a *data* parameter with information in MusicJSON format and a second parameter named *callback*, as the *JSONP* function to be returned. To give an example, Table 3 shows a complete update request to change the duration of note with ref "*note_84*" to a new value of 12.

url	`http://evmusic.fauno.org/storage03/update`
callback	`stcCallback1002`
data	`{"duration":12,"ref":"note_84"}`

Table 3. MusicJSON request parameters

After updating the corresponding data in the storage service, the returned response has the following form:

```
Callback({"message": "Message Content",
"data":[responsed data], "success": true})
```

This means that the returned data are sent back along with a status message confirming the transaction, everything encapsulated in a function with the name of the *callback*, as required by the JSONP data transaction, so the service can accept AJAX requests from other domains. In our example, here is is the response received to the update request above:

```
stcCallback1002({"message": "Updated
record", "data": {"track": "1", "objclass":
"note", "pitch": "39", "start": 6,
"duration": 12, "ref": "note_84", "id":
84}, "success": true}
```

4.3 Library Use

Library services can make use of predefined musical objects. To use a library service, this must be declared in *lib* section of the document in a key-value pair. The reference name for that library is assigned as the key, while the URL that returns the library itself, usually encapsulated as JSONP, is assigned as the value.

```
"lib": {
  "main": "http://evmusic.fauno.org/lib/
main/instruments",
  ... ,
}
```

The content of the library returned from the given address is:

```
LibraryCallBack({
"message":"Sent data: 45 entries",
"data":{
"flute":{
    "objclass":"pinstrument",
    "families":[
        "wind", "air",
        "wood", "woodwind"],
    "clef":"treble",
    "transpose":0,
    "stafforder":1.2,
    "constraints":{
        "minpitch":"c4",
        "maxpitch":"c7",
        "usualpitch":"g5",
        "polyphony":1,
        "maxtime":12,
        "maxspeed":90,
        "maxlegatointerval":17
        },
    "techniques":[
        "frullato", "whistle",
        "air", "keysound"]
    },
"oboe":{
    "objclass":"pinstrument",
    ...
    }
...
},
"success":"true"})
```

In this case, the complete library is returned, but it is also possible to ask for only one object from the library, by completing the URL with the *key* that corresponds to that object. Thus, instead of downloading the whole library, we save memory by downloading only the data we need. For instance, in order to access only the flute instrument, we would use the following URL: `"http://evmusic.fauno.org/lib/main/instruments/flute"`.

In order to use an object from the library, the *x.lib* prefix followed by the library key must be indicated as reference, as shown in Table 2. The MusicJSON code of the referred object will replace the library call during parsing:

```
{
 "objclass": "scoinstrument",
 "name": "Flauta",
 "value": "x.lib.main.flute"
 }
```

4.4 MusicJSON File

Every EvMusic object, from single notes to complex structures, can be serialized into a MusicJSON text and subsequently transmitted through the Web. In addition, MusicJSON can be used as a music format for local storage of compositions. Next listing code shows a draft example of the proposed description of an EvMusic file.

```
{"objclass":"evmusicfile","ver":"1002",
"content":
    {"lib":{

"instruments":"http://evmusic.fauno.org/lib/
main/instruments",
        "pcstypes":
"http://evmusic.fauno.org/lib/main/
pcstypes",
        "mypcs": "http://evmusic.fauno.org/
lib/jesus/pcstypes",
        "mymotives":
"http://evmusic.fauno.org/lib/jesus/motives"
        },
    "def":{
        "ma": {"objclass":"motive",
            "symbol":[  0,7, 5,4,2,0  ],
            "slength": "+-+- +-++ +---"},
        "flamenco": {"objclass":"pcstype",
            "pcs":[  0,5,7,13 ],},
        },
    "orc":{
        "flauta": {"objclass":"instrument",

"value":"x.lib.instruments.flute",
            "role":"r1"}
        "cello": {"objclass":"instrument",

"value":"x.lib.instruments.cello",
            "role":"r2"}
        },
    "score":{
        "objclass": "score",
        "pos": 0,
        "events": [{
            "objclass": "section",
            "pos": 1,
            "events": [{
              "objclass": "staff",
              "name": "flauta",
              "pspellorder": 1,
              "events": [{
                "objclass": "part",
                "track": 1,
                "pos": 1,
                "events": [{
                  "objclass": "note",
                  "pos": 0,
                  "dur": "0.5",
                  "pitch": 62,
                  "art": "st"
                },
                ...

            ... ]},
            {"objclass":"section","pos": 60,
        ...
    },],}}}
}}
```

The code shows four sections in the content. The second section named *lib* is a dictionary of libraries to be loaded. Both main and user libraries can be addressed. The following section includes local definitions of objects. As an example, a *motive* and a *chord type* are defined. Next section establishes instrumentation assignments by means of the arrangement object *role*. Last section is the score itself, where all events are placed in a tree structure using *parts*. Using MusicJSON as the intermediary communication format enables us to connect several music services forming a cloud composition system.

5 . CONCLUSION

This paper puts forward a model of musical representation for the Computer Music Cloud, a new paradigm in which musical computing systems are distributed in several

musical services over a computing cloud. This new environment allows to build new systems by putting various services to work together. We present a new architecture which allows to design specialized autonomous musical services that can be implemented separately in different platforms, and may even serve multiple systems simultaneously.

Effective integration of such musical services in the CMC depends on the definition of a music representation that they all share and that will enable efficient exchange of musical information. Musical representation should fulfill three main requirements:1) to have a high and flexible representativity for music composition, 2) to provide a form allowing music objects to be stored as entries of a database; and 3) to count on a text format that facilitates information exchange, as well as the integration of different data sources.

Our new proposal is based on the robust musical representation for EvMusic composition described in UML which has proved effective in actual composition experiments [1,3]. Above this abstract model of classes, the database form representation and the textual representation MusicJSON have been incorporated.

MusicJSON can represent the complete EvMusic class system together with its different topologies and its comprehensive treatment of musical pitch. In addition to being compatible with conventional musical notation, it can represent higher abstraction structures at multiple levels. It is not only a simple format for exchange of musical objects in text form, but it also integrates musical information from different services. It can also handle references to external objects and libraries. Several examples of use have been shown: representing a music fragment, protocol for sharing musical elements between services, use of libraries and the file format. MusicJSON is based on JSON, an increasingly used format on the Internet. Therefore it inherits its expandability and prototyping features, and benefits from its extensive library of available tools and services.

MusicJSON, EvMusic representation and the database musical format have been tested in real CMC prototypes that incorporate different types of music services [2]. Significantly, it is one of the first proposals for music representation in the new paradigm of Musical Composition in the Cloud. This CMC approach also opens multiple possibilities for derivative work. Once you define an efficient shared music representation for the cloud, any music service can be easily incorporated into the new paradigm: services that translate input and output representations, some applications for collaborative composition among multiple users, musical teaching assistants, and even the integration of true intelligent composition agents. It provides a promising environment for the research in Musical Artificial Intelligence (MAI), where specialised agents can cooperate in a music composition environment sharing the same music representation [4]. Likewise, the paradigm shift that involves the CMC, offers new interesting possibilities for web applications acting as user interfaces in the Computer Music Cloud, taking advantage of new technological developments such as the upcoming HTML5 [14]

REFERENCES

1. Alvaro, J.L., Miranda, E.R., Barros, B. "*Music Knowledge Analysis: Towards an Efficient Representation for Composition*", Current Topics in Artificial Intelligence; LNCS 4177, Springer-Verlag, pp. 331-341(2006)
2. Alvaro, J.L. and Barros, B. "*Composing Music in the Cloud*", Proceedings of the International Symposium on Computer Music Modeling and Retrieval, Málaga 2010
3. Alvaro, J.L. : "*Painting Music with Motives, Twelve Years of Symbolic Pitch Composition*". Under Review
4. Alvaro, J.L. : "*Intelligent Music Clouds*". To be Appeared
5. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. "*Above the Clouds: A Berkeley View of Cloud Computing*" {White Paper} http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.
6. ECMAScript Language Specification, http://www.ecma-international.org/publications/standards/Ecma-262.htm
7. Geelan, J: "*Twenty Experts Define Cloud Computing*", Cloud Computing Journal, SYS-CON Media Inc. (2008) http://cloudcomputing.sys-con.com/node/612375/print
8. Google Apps. http://www.google.com/apps/
9. Ippolito, B. "*Remote JSON - JSONP- JSON with Padding*" (2005) http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/
10. JavaScript. http://en.wikipedia.org/wiki/JavaScript.
11. Introducing JSON: http://www.json.org/
12. Martin, R.C. :"*UML Class Diagrams*" Object Mentor articles (1997) http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf
13. OMG,. Unified Modeling Language: Superstructure. Version 2.1.1, Retrieved from: <http://www.omg.org/uml>, (2007).
14. W3C "*HTML5 A vocabulary and associated APIs for HTML and XHTML*" W3C Editor's Draft http://dev.w3.org/html5/spec/
15. Walter B. Hewlett and Eleanor Selfridge-Field (eds) *"The Virtual Score"*. MIT Press (2001)

STRATEGIES TOWARDS THE AUTOMATIC ANNOTATION OF CLASSICAL PIANO MUSIC

Bernhard Niedermayer[1]
[1]Department for Computational Perception
Johannes Kepler University Linz, Austria
music@jku.at

Gerhard Widmer[1,2]
[2]Austrian Research Institute for
Artificial Intelligence, Vienna, Austria
music@ofai.at

ABSTRACT

Analysis and description of musical expression is a substantial field within musicology. However, manual annotation of large corpora of music, a prerequisite for describing and comparing different artists' styles, is very labor-intensive. Therefore, computer systems are needed that can annotate recordings of different performances automatically, requiring only minimal corrections by the user. In this paper, we apply Dynamic Time Warping for audio-to-score alignment to extract the onset times of all individual notes within an audio recording, and compare two strategies for improving accuracy. The first one is based on increasing the temporal resolution of the features used. To cope with constraints in terms of computational costs, we apply a divide and conquer pattern. The second strategy is introducing a post-processing step in which the onset time of each individual note is revised. The advantage of this method is that, in contrast to default algorithms, arpeggios and asynchronies can be resolved as well.

1. INTRODUCTION

An important subfield of musicology is the analysis and description of musical style and expression. However, large corpora of annotated pieces of music played by several performers are needed to extract meaningful patterns or to support previously developed hypotheses. Such data can be acquired by performing pieces on computer-monitored instruments.

Despite the advantage of providing accurate and extensive data, using computer-monitored instruments for data acquisition has several substantial shortcomings. First of all, one can assume that music students might be persuaded quite easily to take part in such a project, but it will be hard to persuade top-class artists to do so. Secondly, it is not possible to analyze an artist's expressive evolution over long periods. And finally, research could not include artists who, although dead, remain famous and whose music is enjoyed by a broad audience.

Another source of data are audio recordings, which are not only cheap but also available in an extensive variety. However, the raw audio signal must be annotated before any high-level analysis can be performed, and manual annotation is very labor-intensive. In order to carry out research on large corpora of music, automatic – or at least semi-automatic – methods for data acquisition are needed.

The most general approach of collecting symbolic data from audio recordings would be Automatic Music Transcription. However, accuracy and robustness of state-of-the-art transcription methods do not meet the requirements of applications such as musical performance analysis. Especially in the context of classical music, where it can be assumed that the piece played is known in advance, using an audio file in combination with additional information given by the score has therefore become a common practice. Since the notes played are known a priori, the task is to extract the exact parameters of each note from the audio recording.

Such parameters do not only include the timing and the loudness of a note, but also characteristics such as its articulation or, when considering piano, pedal pressure. However, since knowing at which exact point in an audio signal a note is played is a prerequisite for estimating further properties, most current research is focused solely on this. The task is to temporally align or synchronize the notes given by the score to an audio recording - a process known as audio-to-score alignment.

In doing so, features are calculated from individual time frames of the audio signal. There are two main state-of-the art approaches to incorporating the score information: The score, which is by default given in MIDI-format, is used to either build a graphical model [1], such as an HMM, or it is used to compute a sequence of the same features as extracted from the audio signal [2]. Score and audio representations are then related to each other using the Viterbi algorithm or Dynamic Time Warping.

We use Dynamic Time Warping to compute this alignment. Since the algorithm is of quadratic complexity in both time and space, the temporal resolution of the features extracted cannot be increased arbitrarily without encountering limitations in terms of computational cost. One method of reducing the complexity is to apply a *divide and conquer* pattern splitting a piece into several sections using anchor notes, for which the timing is known. Dynamic Time Warping can then be performed on these individual sections without losing generality.

In [3], which originally introduced this approach, anchor notes were selected by the user. We propose a method

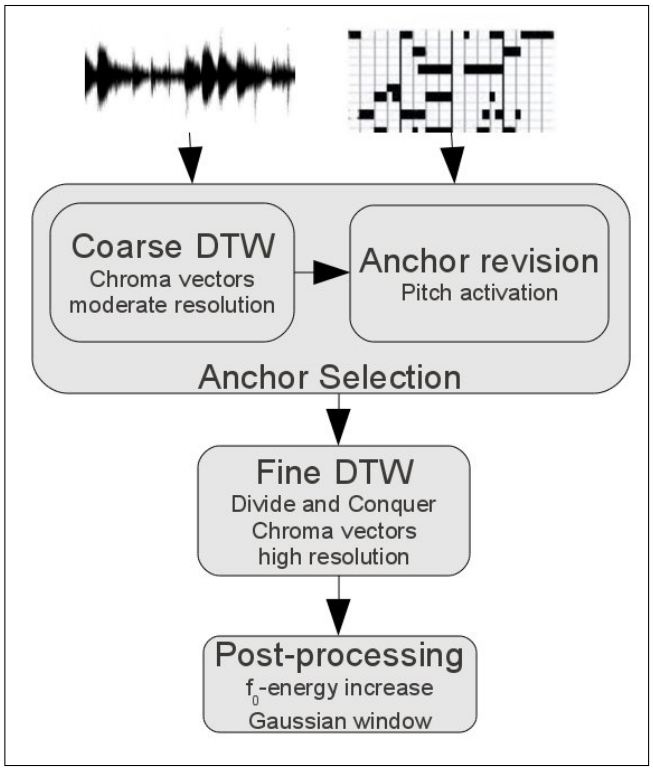


Figure 1. System overview

for extracting such anchor notes automatically. To this end, Dynamic Time Warping is performed once, using a coarse temporal resolution. Based on this initial estimate, anchor notes for which the timing can be extracted with relatively high confidence are identified and their onset times are revised.

Another shortcoming Dynamic Time Warping shares with the approach based upon graphical models is that several notes that occur simultaneously in the score, such as the individual notes of a chord, are always aligned to the same time frame of the audio signal. This is probably not relevant to applications such as augmented audio players. In performance analysis, however, this precludes the handling of arpeggiations or asynchronies. Therefore, we propose a post-processing step in which the onset time of each individual note is revised.

This post-processing step resembles the one described in [4] in both methodology and results. However, [4] used a beta-distribution to model the expectation strength of a note occurring at a certain point between two anchor notes. The beta-distribution was chosen because of its restriction to a fixed interval and the flexibility of its shape, but its use lacks probability-theoretic justification. In this paper, we show that comparable results can be obtained, by applying a weighting which reflects the normally distributed errors made by Dynamic Time Warping.

Figure 1 shows the system architecture, which is further described below as follows. First, we give an overview of related work in Section 2. Then, we explain the audio-to-midi alignment using coarse and fine Dynamic Time Warping in Section 3 and the extraction of the anchor notes based on the coarse alignment in Section 4. Section 5 describes the post-processing step. In Section 6, an evaluation of the system is presented, followed by conclusions in Section 7.

2. RELATED WORK

In offline audio-to-score alignment, a major group of approaches is based on chroma vectors in combination with Dynamic Time Warping (DTW) [2, 3, 4, 5, 6, 7]. This method has proven to yield robust global alignments. However, it cannot compete with onset detection algorithms concerning local accuracy. This was shown in [8], where, as a consequence, the idea of chroma vectors was combined with (pitch-wise) spectral flux – a feature used in onset detection.

A way of applying machine learning techniques to refine music alignments was shown in [5]: A neural network that detects note boundaries is trained on the result of an alignment. In an iterative process, the alignment can then be improved using the neural network's output, and the training is repeated.

Another approach of improving accuracy is to increase temporal resolution. Since DTW is of the order $O(n^2)$, this method is constrained by computational costs. The divide and conquer principle aside, [6] uses a multi-scale algorithm where in each iteration the resolution is increased and, at the same time, the area searched for an optimal alignment is narrowed down.

[7] and [9] combined those two strategies in an efficient way: Both compute an alignment based on DTW and then refine the note onsets within a search window around the initial estimates. Since the size of these search windows is small, a relatively high temporal resolution can be chosen. The additional features used in this post-processing step emphasize onsets of individual pitches. In doing so, the DTW algorithm's problem of unresolved arpeggiations or asynchronies becomes irrelevant. However, in contrast to the system presented here, the method in [9] relies on manual path initialization in the DTW step, and in [7], potentially conflicting notes are not revised, only marked for further processing.

3. AUDIO-TO-MIDI ALIGNMENT

3.1 Chroma Vectors

Chroma vectors are the feature used in most alignment systems because of their robustness to several common phenomena in music, such as changing timbre or different degrees of polyphony. In [2], chroma vectors were shown to outperform several other features in the context of audio matching and alignment. They consist of a 12-dimensional vector per time frame, in which each element represents one pitch class (C, C#, D, ...).

When calculating chroma vectors from a midi file, the energies (in midi terminology *velocities*) of all pitches belonging to the same pitch class are summed up. Additionally, it is beneficial to also consider harmonics by adding decreasing contributions of energies to the corresponding pitch classes. In contrast, when considering an audio sig-

nal, the pitches of notes sounding within a certain time frame are not known a priori. In this case, the values are computed based on an STFT spectrogram by summing up the energies of those frequency bins which are mapped to the same pitch class. The mapping is done by choosing the pitch (and the corresponding pitch class with its index i) with the smallest frequency deviation from a bin's center frequency f_k, according to

$$i = \left[\text{round}\left(12\log_2\left(\frac{f_k}{440}\right)\right)\right] \mod 12 \qquad (1)$$

Within the work reported here, we use two STFT configurations: (i) a window size of 4096 samples and a hop size of 1024 samples, referred to as *moderate resolution*; and (ii) a window size 512 and a hop size 128, referred to as *high resolution*.

3.2 Dynamic Time Warping

After the feature extraction step, the score and the audio signal are both represented by a sequence of feature vectors. To evaluate an alignment, a cost function must be defined which measures the error made when aligning a specific frame of the first sequence to the corresponding frame of the second one. Preliminary experiments showed that the Euclidean distance yields better results within our framework than other functions, such as the cosine distance or the symmetric Kullback-Leibler divergence.

Using this cost function, a similarity matrix S can be calculated. The rows of S represent the time frames of the audio recording, while the columns represent the time frames of the score. Each value S_{ij} gives the cost of aligning frame i of the audio signal to frame j of the score. All continuous and monotonic paths through S which begin and end at the two end-points of the main diagonal represent valid alignments. The sum of all S_{ij} along an alignment path is the respective global alignment cost.

DTW computes an optimal alignment, i.e., the one minimizing the global cost, in two steps. In the first one, the optimal cost C_{ij} of each partial alignment, ending with frame i of the audio signal being aligned to frame j of the score representation, is calculated according to

$$C(i,j) = \min\begin{cases} C(i-1,j-1) + S_{ij} \\ C(i-1,j) + S_{ij} \\ C(i,j-1) + S_{ij} \end{cases} \qquad (2)$$

By starting at $C_{0,0} = S_{0,0}$ and storing all intermediary results in a matrix C, this recursion can be calculated efficiently.

$C_{N-1,M-1}$ is the minimal global alignment cost. However, in this application, the cost itself is not as important as the alignment path corresponding to this optimum. This path is obtained in the second step by backtracking based on knowledge of which of the three options in equation 2 was used in each step. This information can easily be stored during the forward step. For a more detailed description of the basic DTW algorithm, we refer the interested reader to [10].

3.3 Efficiency Considerations

Given two sequences of lengths N and M, DTW is of complexity $O(N * M)$ in both time and space. This resolves to $O(N^2)$ under the assumption that the score is stretched to the length of the audio signal prior to the feature extraction step. This precludes aligning arbitrarily long feature sequences and therefore limits both the lengths of pieces to be aligned and temporal resolution.

A classical method of improving the efficiency of DTW is to constrain the search for an optimal alignment path to a certain area within the similarity matrix S, such as the Itakura parallelogram or the Sakoe-Chiba band [10]. This is based on the assumption that expressive tempo changes will not exceed certain limits, or that the alternation of speeding up and slowing down will prevent the alignment path from deviating from the main diagonal by more than a maximum offset. These approaches can reduce computational costs to the order of $O(2N)$. However, there is the risk that, at some point, the assumptions do not hold and the true alignment path leaves the search area.

Other methods which share similar strengths and weaknesses are Path Pruning, in which only the most promising partial paths with costs below an adaptive threshold are further expanded, Shortcut Path, where only the alignment of frames corresponding to note on- and offsets are considered, and multi-scale DTW [6].

A completely different approach is to perform an online alignment – also known as score following [11]. This algorithm does not consider a piece as a whole, but advances through the audio signal incrementally. This works for arbitrarily long pieces and, leaving the real-time aspect out of consideration, arbitrarily high feature resolutions. The drawback is that the method can only extract instantaneous optima at each step and cannot guarantee that a global optimum is found.

3.4 Divide and Conquer Approach

[3] introduced a divide and conquer approach to improve the efficiency of DTW. Given a set of anchor notes for which the exact timing is known, solving the alignment problem for the whole piece can be reduced to finding optimal alignments between each pair of consecutive anchor notes. Given a maximal interval c between two anchors, the sub-DTWs are computed in $O(c^2)$ in both time and space. When considering the whole piece, the space complexity of $O(c^2)\widehat{=}O(1)$ does not change. Time complexity, however, increases to $O(c^2 * N/c) = O(c * N)\widehat{=}O(N)$. Compared to the original algorithm of order $O(N * M)$, this approach reduces complexity and guarantees that a globally optimal alignment is found.

This increase in efficiency is countered by the additional problem of how to identify suitable anchor notes and how to extract their respective onset times. [3] proposed an approach in which the user selects an anchor configuration manually or verifies suggestions made by the algorithm. These suggestions are established based on cues such as pauses, long isolated fortissimo chords, or notes with salient fundamental pitches, i.e., pitches that do not overlap with harmonics of concurrently played notes.

4. ANCHOR EXTRACTION

In this paper, we show how anchor notes can be determined automatically. The selection is based on a coarse DTW computed as described above. Within a search window around the first onset estimate of each note, a revised candidate is then extracted using the Pitch Activation feature as described in [7]. Finally, all notes for which ambiguities arise are dropped from the list of anchors.

Doing this has two implications. First, basing anchor selection on an initial alignment alters the method, shifting it away from the original divide and conquer approach and towards a special multi-scale DTW. Second, the algorithm is not guaranteed to find the global optimum, since errors in the anchor selection result in inaccurate alignments.

4.1 Pitch Activation

The feature used for revising onset candidates is pitch activation, which is calculated by applying a modification of non-negative matrix factorization (NMF) to audio data in the frequency domain. NMF is the decomposition of an input matrix V of size $n \times m$ into two output matrices W and H of sizes $m \times r$ and $r \times n$ such that the elements of all these matrices are non-negative and

$$V \approx WH \tag{3}$$

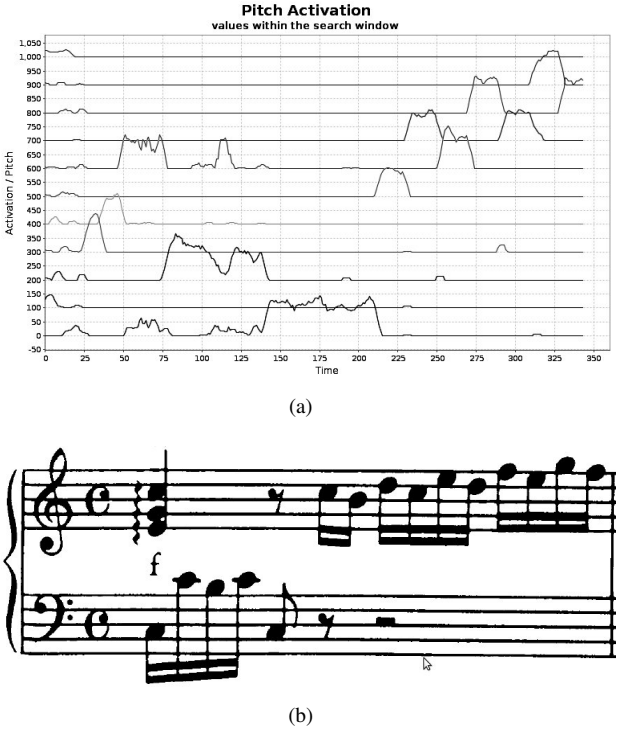
The reconstruction error, i.e., the deviation of WH from V, can be measured with several cost functions such as the Euclidean distance or the Kullback-Leibler divergence. An optimal factorization is calculated by minimizing this cost.

Applied to audio processing, NMF can be used to factorize a spectrogram into a dictionary W of weighted frequency groups and the corresponding activation energies H of these frequency groups over time. Depending on V and the parameter r, the base components in W can represent models of single tones or chords. But since the NMF algorithm, as originally introduced in [12], is unsupervised, it is more likely that some of the components also describe single partials, special patterns during an attack, sustain, or decay phase of a note, or even just noise. However, in the context of audio-to-score alignment, where the piece played is known a priori, we assume the instrument or set of instruments playing to be known as well. Thus, tone models can also be trained using supervised methods.

Based on this assumption, a dictionary W of tone models is trained in advance ([7, 4, 13]). The training data comprises recordings of single tones played at several degrees of loudness on the instrument under consideration. A short-time Fourier transform is calculated and factorized while exploiting knowledge of the tone samples. Since only one tone is played in each sample, the number of components r is set to one. The activation energy of this component w over time is fixed to $\overline{h}$ and assumed to be equal to the amplitude envelope. Equation 3 then resolves to

$$V \approx w\overline{h} \tag{4}$$

By minimizing the reconstruction error, a tone model is learned from each training sample. Since the relative energy of the harmonics depends on the intensity, preliminary models are trained using several degrees of loudness, and the final model is then obtained by taking the average weight for each frequency.



(a)

(b)

Figure 2. Example of activation patterns: (b) shows the first bar of Mozart's piano sonata k279. In (a) the activation patterns of the single pitches are plotted in ascending order.

Given this fixed dictionary $\overline{W}$, equation 3 can be rewritten as

$$v \approx \overline{W}h \tag{5}$$

where v and h are single columns of V and H that can now be processed independently. Since in both equation 5 and equation 4 there is only one variable left, a non-negative least squares optimization minimizing the mean square criterion

$$f = \frac{1}{2} \parallel \overline{W}h - v \parallel^2 \tag{6}$$

can be applied instead of the original NMF methods. This not only reduces computational costs but also, due to the independence of individual frames, makes the pitch activation h – a frame-wise f_0 estimation – a feature suitable for online algorithms.

Figure 2 shows an example of such activation patterns. The dictionary used for the factorization consisted of the models describing those pitches which are expected to be played within the time range shown only.

4.2 Anchor selection

For the anchor selection, the pitch activation feature is used to revise the onset estimates obtained by DTW. $\overline{W}$ is composed from the tone models of those pitches, expected to

be played within the search window and an additional component modeling white noise. The new onset candidate is set to the frame with the maximal increase of h_p, where p is the index of the component corresponding to the pitch of the note under consideration. In cases in which the onset is unambiguous, these onset candidates have proven to be very accurate. But in cases in which the same note is repeated several times within the search window, this method is too simple and very likely to fail.

To solve this dilemma, notes that are expected to be played more often than once within the search window are disregarded as potential anchors. Also, all notes for which the onset is ambiguous, i.e., for which the difference between the onset estimate obtained by the initial coarse DTW and the revised onset time exceeds a certain threshold, are dropped from the list of anchor candidates. In doing so, the anchor selection benefits from the robustness of the DTW as well as from the accuracy of the pitch activation-based onset revision.

Although this approach is very simple, our evaluation in Section 6 shows that by adjusting the onset times of these anchor notes only, the overall result is improved significantly.

5. POST PROCESSING FOR POLYPHONIC PIECES

As pointed out before, both DTW and alignment methods based on graphical models suffer from the shortcoming that notes which are indicated in the score as being played simultaneously simultaneously will inherently be aligned to the same time frame within the audio signal. Hence, increasing the temporal feature resolution to arbitrary dimensions as described in Section 3 benefits monophonic pieces for which the onsets of individual notes are extracted and pieces which are too long to be processed as a whole, even when using moderate resolutions. However, when considering polyphonic pieces, notes which are indicated in the score as being played simultaneously will never be played precisely simultaneously by the performer due to arpeggiations or asynchronies. Therefore, using a resolution high enough to break a chord down into several notes and their individual onsets results in an ambiguous onset time for the chord as a whole. It is not clear if the estimate obtained by DTW or the Viterbi algorithm represents the note which is played first, the one which is played last, or some time in between where the cumulative energy of all chord notes has exceeded a certain threshold.

To overcome this problem, we apply a post-processing step in which the onset times of all non-anchor notes are revised as well. We assume that the high-resolution DTW computed by our system yields relatively accurate estimations and that deviations from the real onset times follow a normal distribution. Therefore, on the one hand, a search window of length $2l$ centered around the initial estimate is considered. On the other hand, feature values computed to refine the onset time are weighted using a Gaussian window.

However, the choice of features is not trivial. Pure onset detection functions, such as spectral flux, are not sufficient, since, when dealing with polyphonic pieces, the onsets of other chord notes must be expected to occur within the immediate vicinity of a note. Also, the pitch activation feature used for anchor selection is not suitable, since it performs poorly in situations of repeated notes.

Preliminary experiments showed that, considering the remaining non-anchor notes, the increase in the energy of the fundamental frequency of a note is the most reliable and accurate onset estimate. We obtain this information from a constant Q transform with a frequency resolution of one bin per semitone and set the revised onset candidate to the time frame at which the maximal increase occurs.

Parameter values of around 100 ms for the search radius l and 0.4 for the standard deviation σ of the Gaussian window have proven to yield good results. A detailed evaluation can be found in the next section.

6. EVALUATION

6.1 Evaluation Method

The evaluation was done using the first movements of 11 Mozart sonatas played by a professional pianist. The performances were recorded on a computer-monitored Bösendorfer SE290 grand piano, logging the exact onset times of all notes. The data comprises more than 30.000 notes with an overall performance time of more than one hour. Scores were presented to the system in midi format.

The absolute temporal displacement between aligned notes and the ground truth served as the main evaluation criterion. We investigated the median absolute displacement, the 75^{th}, and the 95^{th} percentile. In our opinion, this shows a clearer view of a system's performance than the mean and variance of absolute displacements, since these values are more sensitive to outliers. When considering only mean and variance, it is difficult to distinguish systems that yield accurate estimates for most notes but produce a few outliers with relatively large temporal displacement, from systems which are more robust but less accurate.

In the evaluation of the whole system including the post-processing, we include two other criteria. The long-term goal of our research is to provide an annotation system that detects onset times as accurately as a human. Only a very small number of notes for which manual correction is needed should remain. [14] showed that humans do not perceive timing deviations smaller than 10 ms. Therefore, we also investigated the proportion of notes which are aligned with a timing deviation below this threshold.

Furthermore, we determined the percentage of the notes aligned with a displacement of less than 50 ms. This criterion is well known from the field of onset detection. Here, it reflects the ratio of reasonably well aligned notes to outliers.

6.2 Evaluation Results

Table 1 shows the accuracy of the selected anchor notes in comparison to the non-anchor notes before and after performing the fine resolution DTW. One can clearly see that the anchor nodes are indeed more accurate than the

piece	duration	# notes	# anchors	$50\% < x$[ms]			$75\% < x$[ms]			$95\% < x$[ms]		
				anch.	orig.	new	anch.	orig.	new	anch.	orig.	new
K.279-1	4:55	2803	885	6.0	15.7	15.8	13.3	29.6	29.8	43.7	127	128
K.280-1	4:48	2491	987	5.7	23.2	22.9	12.1	44.6	44.6	43.5	165	165
K.281-1	4:29	2648	954	6.6	25.2	25.1	13.3	47.3	47.2	47.8	137	138
K.282-1	7:35	1907	513	7.8	26.8	26.7	14.5	64.0	64.2	80.8	388	389
K.283-1	5:22	3304	875	8.1	15.5	15.4	14.4	27.8	27.8	40.9	67.8	68.2
K.284-1	5:17	3700	853	7.0	15.2	15.3	15.9	30.6	30.7	62.3	108	107
K.330-1	6:14	3160	888	6.0	16.0	15.9	10.5	29.4	29.3	37.9	148	146
K.332-1	6:02	3470	844	11.5	22.8	22.9	19.0	42.1	42.3	61.3	167	168
K.333-1	6:44	3774	1122	8.0	17.1	17.1	14.4	30.3	30.4	42.1	105	105
K.457-1	6:15	2993	885	9.2	21.3	21.3	16.3	40.5	40.3	59.8	267	267
K.475-1	4:58	1284	371	15.4	36.3	36.3	23.7	92.0	92.5	79.4	270	270

Table 1. Comparison between accuracy (median, 75^{th} percentile, and 95^{th} percentile) of the anchor notes (anch.), the non-anchor notes computed by the coarse DTW (orig.), and the non-anchor notes after performing the fine DTW implementing the divide and conquer pattern (new)

piece	# notes	$50\% < x$[ms]			$75\% < x$[ms]			$95\% < x$[ms]		
		orig.	anch.	new	orig.	anch.	new	orig.	anch.	new
K.279-1	2803	15.7	11.2	11.8	30.0	24.5	25.7	103	101	103
K.280-1	2491	23.6	12.7	13.4	41.9	32.0	32.8	126	127	126
K.281-1	2648	24.2	15.2	16.1	42.4	36.5	36.9	114	114	114
K.282-1	1907	23.5	18.7	19.6	53.7	47.2	48.1	354	354	354
K.283-1	3304	14.6	12.7	12.8	27.1	24.5	24.8	62.0	60.8	60.9
K.284-1	3700	15.4	12.5	13.1	31.0	26.9	27.4	96.8	98.0	98.0
K.330-1	3160	14.9	11.4	11.8	27.7	24.0	24.7	118	118	115
K.332-1	3470	20.5	18.4	18.6	38.6	35.6	36.3	138	140	138
K.333-1	3774	16.2	12.9	13.4	29.3	25.8	26.4	79.6	75.8	76.4
K.457-1	2993	19.4	15.7	16.2	36.9	33.5	34.2	204	203	202
K.475-1	1284	29.7	24.5	25.0	68.4	65.5	65.9	224	224	224
all	31534	18.4	14.1	14.7	35.2	30.1	30.8	130	131	131

Table 2. Overall accuracy (median, 75^{th} percentile, and 95^{th} percentile) of the divide and conquer DTW (new) compared to the coarse DTW with the anchor notes revised (anch.) and the coarse DTW without anchor note revisions (orig.)

remaining notes. However, it is remarkable that the high-resolution DTW implementing the divide and conquer principle does not improve the results obtained by the original implementation using a moderate temporal resolution. A discussion on this issue is given in the next section.

It is worth mentioning that, due to the semi-automatic nature of the anchor selection, only a very small number of anchors was used in [3]. In our approach, the number of anchors is much larger, as shown in Table 1. Qualitative analysis of single passages showed that there are "easy" sections, in which no ambiguities occur and almost every single note is chosen to serve as anchor, whereas there are "difficult" sections within a piece in which only few anchors are found. Although not required, the high number of anchor notes is desirable, since, in contrast to [3], the objective here was not only efficiency, but also to investigate accuracy aspects. This approach to anchor selection clearly outperforms the DTW variant in terms of accuracy.

Recalling the whole system's architecture, as depicted in Figure 1, Table 2 compares the results after the individual stages - the coarse DTW, the coarse DTW with revised anchor notes, and the high-resolution DTW exploiting these anchor notes. It is even more apparent that, while the revision of anchor notes improves the result significantly, the additional high-resolution DTW even decreases the overall accuracy very slightly.

The overall accuracy of the whole system including the post-processing step is listed in Table 3. According to our evaluation criteria, more than 90% of all notes were aligned reasonably well, i.e., such that evaluation frameworks used in onset detection would classify them as correct. Almost half of the notes were aligned with an error small enough not to be perceived by a human listener. Comparing the percentiles to the ones given in Table 2 clearly proves the benefit of the post-processing step.

Since the high-resolution DTW did not improve the results, the question arises if the system performed better without this step. Applying the post-processing method directly to the results of the anchor selection yielded similar results as the whole system. The overall number of notes with a temporal displacement of less than 10 ms increased slightly to 49.2%, while the number of notes with an alignment error of less than 50 ms decreased to 88.9%.

piece	# notes	$50\% < x$[ms]	$75\% < x$[ms]	$95\% < x$[ms]	$x < 10\,ms$	$x < 50\,ms$
K.279-1	2803	7.7	20.3	93.3	59.4%	90.7%
K.280-1	2491	7.3	16.0	79.0	62.0%	91.5%
K.281-1	2648	9.1	21.8	112	53.4%	89.8%
K.282-1	1907	11.4	22.0	258	44.3%	85.9%
K.283-1	3304	10.1	17.6	51.7	49.3%	94.8%
K.284-1	3700	8.1	20.1	78.5	57.7%	90.4%
K.330-1	3160	8.0	16.0	66.3	58.8%	93.5%
K.332-1	3470	15.8	25.8	106	31.6%	90.0%
K.333-1	3774	10.4	19.0	60.3	48.5%	93.3%
K.457-1	2993	13.4	25.1	164	37.6%	86.1%
K.475-1	1284	19.0	30.0	359	24.7%	85.6%
all	31534	10.3	21.3	92.6	49.0%	90.7%

Table 3. Overall accuracy after post-processing

7. CONCLUSIONS

We have described two strategies to improve the accuracy of offline audio-to-score alignments. One is to apply a higher feature resolution. In order not to be constrained by computational costs, a divide and conquer approach exploiting selected anchor notes was used.

The second strategy is to include a post-processing step which works on the level of individual notes. Here, we have proposed an approach that combines the robustness of DTW with the accuracy of a special onset feature by weighting the feature values using a Gaussian window centered around the onset estimate obtained by DTW. In [4], which introduced a very similar post-processing method, the analog weighting of feature values was done based on a beta-distribution, which was used for pragmatic reasons only. In contrast, the Gaussian window applied in our approach is justified by the actual data.

Our evaluation showed that the largest improvement is due to the revision of the anchor notes. Based on this step, increasing the temporal resolution does, remarkably, not yield higher, but even slightly lower overall accuracy. A possible explanation is that, on the one hand, errors caused by arpeggiations or asynchronies cannot be eliminated by DTW or related algorithms, independently of the temporal resolution. On the other hand, the same features – chroma vectors – were used for the initial coarse alignment and the high-resolution DTW. In cases in which chroma vectors, despite their advantages, are not powerful enough to represent all information that would be needed, the feature resolution becomes irrelevant.

Also, only notes for which the revised onset obtained by the pitch activation feature was near the initial estimate were chosen as anchors. This was necessary to exclude ambiguous cases. However, the revised anchors themselves never deviate from the initial alignment path by more than a small threshold. Therefore, the additional information produced by theses corrections is limited.

The post-processing step, in contrast, improved the result of the DTW including the revision of anchor notes significantly. We attribute this to the same factors as mentioned above. First, a new feature which is independent of the ones used previously is introduced to the system and therefore adds new information. Also, since the post-processing steps work at the level of independent notes, asynchronies can now be resolved.

We conclude that the DTW algorithm using features of moderate resolution works with high robustness and satisfactory accuracy. Improvements of the algorithm which can exploit features with higher temporal resolution did not improve the overall results. We will therefore concentrate our future work on more advanced post-processing methods, since this is the area where we see the largest potential for improvements.

8. ACKNOWLEDGEMENTS

This research is supported by the Austrian Federal Ministry for Transport, Innovation and Technology, and the Austrian Science Fund (FWF) under project numbers TRP 109-N23, P19349-N15, and Z159.

9. REFERENCES

[1] C. Raphael: "Aligning Music Audio with Symbolic Scores Using a Hybrid Graphical Model", *Machine Learning*, Vol. 65 (2-3), pp. 389–409, 2006.

[2] N. Hu, R. B. Dannenberg, and G. Tzanetakis: "Polyphonic Audio Matching and Alignment for Music Retrieval", *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, 2003.

[3] M.Müller, F. Kurth and T. Röder: "Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization", *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, 2004.

[4] B. Niedermayer and G. Widmer: "A Multi-Pass Algorithm for Accurate Audio-to-Score Alignment", *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, Utrecht, 2010.

[5] N. Hu and R. B. Dannenberg: "A Bootstrap Method for Training an Accurate Audio Segmenter", *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, 2005.

[6] M. Müller, H. Mattes, and F. Kurth: "An Efficient Multiscale Approach to Audio Synchronization", *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, 2006.

[7] B. Niedermayer: "Improving Accuracy of Polyphonic Music-to-Score Alignment", *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, Kobe, 2009.

[8] S. Ewert and M. Müller: "Refinement Strategies for Music Synchronization", *Proceedings of the 5th International Symposium on Computer Music Modeling and Retrieval (CMMR 2008)* Copenhagen, 2008.

[9] Y. Meron and K. Hirose: "Automatic alignment of a musical score to performed music", *Acoustical Science and Technology*, Vol. 22, No. 3, pp. 189–198, 2001.

[10] Rabiner, L. R. and Juang, B.-H. "Fundamentals of speech recognition". Prentice Hall, Englewood Cliffs, NJ, 1993.

[11] S. Dixon: "Live Tracking of Musical Performances Using On-Line Time Warping", *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx)*, Madrid, 2005.

[12] Lee, D. D. and Seung, H. S. "Algorithms for Non-Negative Matrix Factorization", *Neural Information Processing Systems*. 2000.

[13] F. Sha and L. Saul: "Real-time pitch determination of one or more voices by nonnegative matrix factorization", *Advances in Neural Information Processing Systems 17*, K. Saul, Y. Weiss, and L. Bottou (eds.), MIT Press, Cambridge, MA, 2005.

[14] A. Friberg and J. Sundberg: "Perception of just noticeable time displacement of a tone presented in a metrical sequence at different tempos", *Proceedings of the Stockholm Music Acoustics Conference*, pp. 39–43, Stockholm, 1993.

AUTOMATIC MUSIC TAG CLASSIFICATION BASED ON BLOCK-LEVEL FEATURES

Klaus Seyerlehner[1], **Gerhard Widmer**[1,2], **Markus Schedl**[1], **Peter Knees**[1]
[1]Dept. of Computational Perception, Johannes Kepler University, Linz, Austria
[2]Austrian Research Institute for AI, Vienna, Austria
klaus.seyerlehner@jku.at

ABSTRACT

In this paper we propose to use a set of block-level audio features for automatic tag prediction. As the proposed feature set is extremely high-dimensional we will investigate the Principal Component Analysis (PCA) as compression method to make the tag classification computationally tractable. We will then compare this block-level feature set to a standard feature set that is used in a state-of-the-art tag prediction approach. To compare the two feature sets we report on the tag classification results obtained for two publicly available tag classification datasets using the same classification approach for both feature sets. We will show that the proposed features set outperform the standard feature set, thus contributing to the state-of-the-art in automatic tag prediction.

1. INTRODUCTION

Today there exist many online music platforms like for example Last.fm [1] that allow users to annotate the songs they are listening to with semantic labels, so called *tags*. This way the users themselves collaboratively create semantic descriptions of the available music universe. The tags associated with a song can then for example be used to search for new music (*tag-based browsing*) or to automatically generate music recommendations. One major drawback of tag-based browsing or recommendation systems is that in the case a song has not yet been annotated by a number of users too little or unreliable information is available about a song, such that it cannot be included in the search or recommendation process. This issues is known as the cold-start problem [1].

One approach to solve the cold-start problem for tag-based music search and recommendation systems is to predict tags that users would associate with a given song from the audio signal itself. This task is called *automatic tag prediction* and is a relatively new research area in Music Information Retrieval (MIR). Automatic tag prediction can be interpreted as a special case of multi-label classification. The task of tag prediction can be defined as follows: Given a set of tags $T = \{t_1, ..., t_A\}$ and a set of songs $S = \{s_1, ..., s_R\}$ predict for each song $s_j \in S$ the tag annotation vector $y = (y_1, ..., y_A)$, where $y_i > 0$ if tag t_i has been associated with the audio track by a number of users, and $y_i = 0$ otherwise. Thus, the y_i's describe the strength of the semantic association between a tag t_i and a song s_j and are called *tag affinities* or *semantic weights*. If the semantic weights are mapped to $\{0, 1\}$, then they can be interpreted as class labels. Although tag affinities can be quite valuable in some applications, e.g. automatic similarity estimation [2] or specific retrieval tasks, in this paper we focus on the binary *tag classification* task, which can be interpreted as a specific sub-task of tag prediction, where a tag is either applicable for a given song or not.

In contrast to recent research on automatic tag prediction, which basically focuses on improving the tag classification approach [3, 4], in this paper we propose the use of a new, more powerful set of audio features, namely block-level features. Block-level feature have already proven to be useful in automatic music genre classification [5] and in automatic music similarity estimation [6]. Here we will investigate if block-level features are also useful with respect to the task of automatic tag classification. A specific problem in this context that we will address is the high dimensionality of the described set of block-level features.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 describes the two features sets – block-level features and 'standard' feature set – that are compared in the final evaluation. In section 4 we present the tag classification approach used in the evaluation. Section 5 presents the evaluation datasets, the performance measures, and the results of the conducted tag classification experiments. Conclusions and directions for future work are given in Section 6.

2. RELATED WORK

Although automatic tag predictions is a relatively new area in Music Information Retrieval the ***M**usic **I**nformation **R**etrieval **E**valuation e**X**change* (MIREX) [2], a competitive evaluation, has driven the development of several automatic tag classification systems. A good overview of state-of-the-art systems can therefore be found in the accompanying descriptions of the participating systems in the

[1] www.last.fm

[2] http://www.music-ir.org/mirexwiki

MIREX tag classification task. In the literature, in contrast, there exist only a few publications focusing on automatic tag prediction.

One of the most important contributions to the area of automatic tag prediction is the work of Turnbull et al. [7]. To the best of our knowledge, they proposed the first tag prediction system based on a generative probabilistic model, where each tag is modeled as a distribution over the audio feature space (Delta-MFCC vectors). Furthermore, they also contributed to the evaluation of tag prediction systems by creating an evaluation dataset, the CAL500 dataset (see Section 5.1). Another probabilistic approach was proposed by Hoffman et al. [8]. Their Codeword Bernoulli Average (CBA) model is a probabilistic generative latent variable model. Vector-quantized Delta-MFCCs serve as observations to the generative model. Their approach is a simple and fast and, according to Hoffman et al., outperforms the method of Turnbull et al. on the CAL500 dataset.

Besides these probabilistic models, there are two recent publications by Mahieux et al. [4] and Ness et al. [3] on systems using a stacked hierarchy of binary classifiers. In both of these there is one binary classifier per tag, at the first level. Then the probabilistic output, the predicted tag affinity, of each binary tag classifier is used as the input to a second classification stage, where there is once more one classifier per tag. The advantage of this setup is that the second stage classifiers can now take into account the correlations between tags. Implausible combinations of tag predictions from the first stage can be corrected. We will denote such a classification approach as *"stacked generalization"*, in accordance with [3]. For our feature set comparison we will use exactly the same stacked generalization approach as proposed in [3], as this procedure is already implemented and publicly available via the MARSYAS (Music Analysis, Retrieval and Synthesis for Audio Signals) [3] open source framework. That not only permits us to compare two feature sets using one and the same state-of-the-art tag classification approach, but also to put the obtained results into the context of the MIREX tag classification task: in last year's (2009) run of the tag classification contest the approach based of the MARSYAS framework ranked second (with respect to the per tag f-score), only insignificantly worse than the leading algorithm.

3. FEATURES FOR AUTOMATIC TAG CLASSIFICATION

In this section we first describe the block processing framework (3.1) and then the block-level features (3.2) that are used for automatic tag prediction. This feature set significantly differs from standard feature sets used in music information retrieval and has recently been proposed for automatic genre classification [5] and for content-based music similarity estimation [6]. Unfortunately, the presented block-level are all very high-dimensional, which is not desirable in the context of classification because of the high computational costs resulting from the very high-dimensional feature space. Therefore in subsection 3.3 we propose to compress the block-level audio features via the well-known Principal Component Analysis (PCA). In subsection 3.4 we will then present the standard feature set extracted by the MARSYAS framework that we compare the proposed features to.

[3] www.marsyas.info

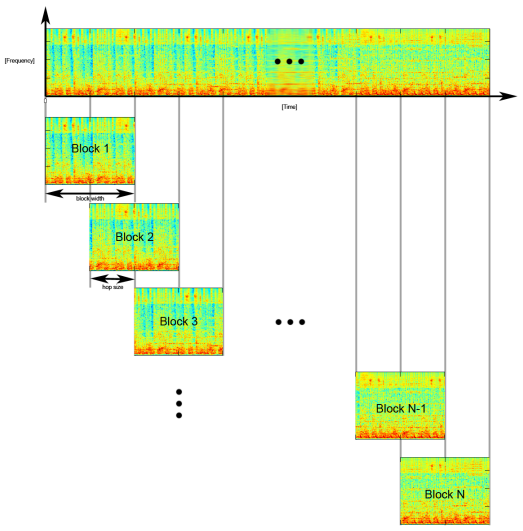


Figure 1. *Block by block processing of the cent spectrum.*

3.1 The block processing Framework

The idea of processing audio block by block is inspired by the feature extraction process described in [9, 10, 11]. However, instead of just computing rhythm patterns or fluctuation patterns from the audio signal one can generalize this approach and define a generic framework. Based on this framework one can then compute other features (e.g., the ones presented in section 3.2) to describe the content of an audio signal. One advantage of block-level features over frame-level features like, e.g., MFCCs is that each block comprises a sequence of several frames, which allows the extracted features to better capture temporal information. The basic block processing framework can be subdivided into two stages: first, the *block processing stage* and second, the *generalization step*.

3.1.1 Block Processing

For block-based audio features the whole spectrum is processed in terms of blocks. Each block consists of a fixed number of spectral frames defined by the block size. Two successive blocks are related by advancing in time by a given number of frames specified by the hop size. Depending on the hop size blocks may overlap, or there can even be unprocessed frames in between the blocks. Although the hop size could also vary within a single file to reduce aliasing effects, here we only consider constant hop sizes. Figure 1 illustrates the basic process.

A block can be interpreted as a matrix that has W columns defined by the block width and H rows defined by the frequency resolution (the number of frequency bins):

$$\text{block} = \begin{bmatrix} b_{H,1} & \cdots & b_{H,W} \\ \vdots & \ddots & \vdots \\ b_{1,1} & \cdots & b_{1,W} \end{bmatrix} \quad (1)$$

3.1.2 Generalization

To come up with a global feature vector per song, the feature values of all blocks must be combined into a single representation for the whole song. To combine local block-level feature values into a model of a song, a summarization function is applied to each dimension of the feature vectors. Typical summarization functions are, for example, the mean, median, certain percentiles, or the variance over a feature dimension. Interestingly, also the classic Bag of Frames approach (BOF) [12] can be interpreted as a special case within this framework. The block size would in this case correspond to a single frame only, and a Gaussian Mixture Model would be used as summarization function. However, we do not consider distribution models as summarization functions here, as our goal is to define a song model whose components can be interpreted as vectors in a vector space. The generalization process is illustrated in Fig. 2 for the median as summarization function.

In the following, we describe how to compute the feature values on a single block and give the specific summarization function for each feature. While Fig. 2 depicts the block level features as vectors, the features described below will be matrices. This makes no difference to the generalization step, however, as the summarization function is applied component by component; the generalized song-level features will thus also be matrices.

3.2 Block-Level Features

3.2.1 Audio Preprocessing

All block-level features presented in this paper are based on the same spectral representation: the cent-scaled magnitude spectrum. To obtain this, the input signal is downsampled to 22 kHz and transformed to the frequency domain by applying a Short Time Fourier Transform (STFT) using a window size of 2048 samples, a hop size of 512 samples and a Hanning window. Then we compute the magnitude spectrum $|X(f)|$ thereof and account for the musical nature of the audio signals by mapping the magnitude spectrum with linear frequency resolution onto the logarithmic Cent scale [13] given by Equation (2).

$$f_{\text{cent}} = 1200 \log_2(f_{\text{Hz}}/(440 * (\sqrt[1200]{2})^{-5700})) \quad (2)$$

The compressed magnitude spectrum $X(k)$ is then transformed according to Eq.3 to obtain a logarithmic scale. Altogether, the mapping onto the Cent scale is a fast approximation of a constant-Q transform, but with constant window size for all frequency bins.

$$X(k)_{dB} = 20 \log_{10}(X(k)) \quad (3)$$

Finally, to make the obtained spectrum loudness-invariant, we normalize it by removing the mean computed over a sliding window from each audio frame as described in [5]. All features presented in the next section are based on the normalized cent spectrum. Note that the reported parameter settings for the audio features in the following subsections were obtained via optimization with respect to a genre classification task (on a different music collection that the ones used here).

3.2.2 Spectral Pattern (SP)

To characterize the frequency or timbral content of each song we take short blocks of the cent spectrum containing 10 frames. A hop size of 5 frames is used. Then we simply sort each frequency band of the block.

$$\text{SP} = \begin{bmatrix} \text{sort}(b_{H,1} & \cdots & b_{H,W}) \\ \vdots & \ddots & \vdots \\ \text{sort}(b_{1,1} & \cdots & b_{1,W}) \end{bmatrix} \quad (4)$$

As summarization function the 0.9 percentile is used.

3.2.3 Delta Spectral Pattern (DSP)

The Delta Spectral Pattern is extracted by computing the difference between the original cent spectrum and a copy of the spectrum delayed by 3 frames, to emphasize onsets. The resulting delta spectrum is rectified so that only positive values are kept. Then we proceed exactly as for the Spectral Pattern and sort each frequency band of a block. A block size of 25 frames and a hop size of 5 frames are used, and the 0.9 percentile serves as summarization function. It is important to note that the DSP's block size differs from the block size of the SP; both were obtained via optimization. Consequently, the SP and the DSP capture information over different time spans.

3.2.4 Variance Delta Spectral Pattern (VDSP)

The feature extraction process of the Variance Delta Spectral Pattern is the same as for the Delta Spectral Pattern (DSP). The only difference is that the Variance is used as summarization function over the individual feature dimensions. While the Delta Spectral Pattern (DSP) tries to capture the strength of onsets, the VDSP should indicate if the strength of the onsets varies over time or, to be more precise, over the individual blocks. A hop size of 5 and a block size of 25 frames are used.

3.2.5 Logarithmic Fluctuation Pattern (LFP)

To represent the rhythmic structure of a song we extract the Logarithmic Fluctuation Patterns, a modified version of the Fluctuation Pattern proposed by Pampalk et al. [9]. A block size of 512 and a hop size of 128 are used. We take the FFT for each frequency band of the block to extract the periodicities in each band. We only keep the amplitude modulations up to 600 bpm. The amplitude modulation coefficients are weighted based on the psychoacoustic model of the fluctuation strength according to the original approach in [9]. To represent the extracted rhythm pattern in a more tempo invariant way, we then follow the idea in [14, 15] and represent periodicity in log scale instead of linear scale. Finally, we blur the resulting pattern with a

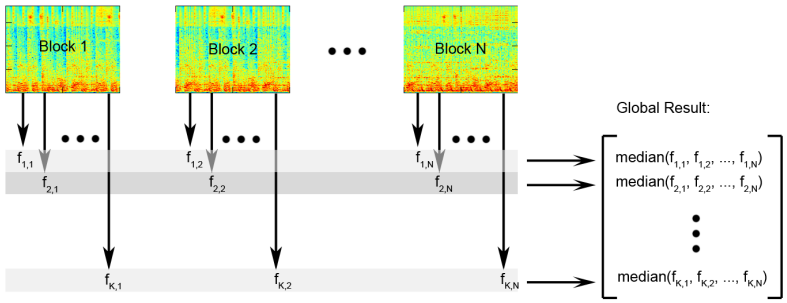


Figure 2. *Generalization from block level features to song feature vectors, with the median as summarization function.*

Gaussian filter, but for the frequency dimension only. The summarization function is the 0.6 percentile.

3.2.6 Correlation Pattern (CP)

To extract the Correlation Pattern the frequency resolution is first reduced to 52 bands. This was found to be useful by optimization and also reduces the dimensionality of the resulting pattern. Then we compute the pairwise linear correlation coefficient (Pearson Correlation) between each pair of frequency bands, which gives a a symmetric correlation matrix. The Correlation Pattern can capture, for example, harmonic relations of frequency bands when sustained musical tones are present. Also rhythmic relations can be reflected by the CP. For example, if a bass drum is always hit simultaneously with a high-hat this would result in a strong positive correlation between low and high frequency bands. Visualizations of the CP show interesting patterns for different types of songs. For example the presence of a singing voice leads to very specific correlation patterns. A block size of 256 frames and a hop size of 128 frames is used. The summarization function for this feature is the 0.5 percentile.

3.2.7 Spectral Contrast Pattern (SCP)

The Spectral Contrast [16] is a feature that roughly estimates the *"tone-ness"* of a spectral frame. This is realized by computing the difference between spectral peaks and valleys in several sub-bands. As strong spectral peaks roughly correspond to tonal components and flat spectral excerpts are often related to noise-like or percussive elements, the difference between peaks and valleys characterizes the toneness in each sub-band. In our implementation the Spectral Contrast is computed from a cent scaled spectrum subdivided into 20 frequency bands. For each audio frame, we compute in each band the difference between the maximum value and the minimum value of the frequency bins within the band. This results in 20 Spectral Contrast values per frame. The values pertaining to an entire block are then sorted within each frequency band, as already described for the SP above. A block size of 40 frames and a hop size of 20 frames are used. The summarization function is the 0.1 percentile.

Figure 3 visualizes the proposed set of features for two different songs, a Hip-Hop and a Jazz song.

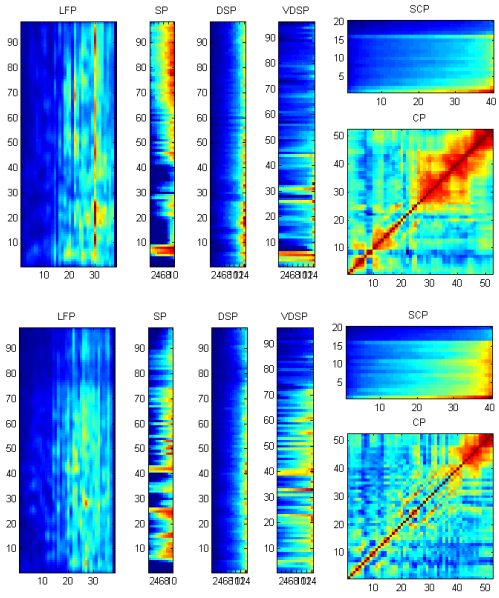


Figure 3. *Visualization of the proposed block-level patterns for a Hip-Hop song (upper) and a Jazz song (lower).*

3.3 PCA Compression

Unfortunately, the described block-level features are all high-dimensional. For instance, an LFP has 37 (periodicities) $\times$ 98 (audio frequencies) $=$ 3626 dimensions. Feature spaces of such high dimensionality are a serious problem in classification tasks, in terms of both overfitting and computational complexity. However, thanks to the vector space representation of the individual features we can use standard dimensionality reduction techniques to reduce the size of the features. A standard Principal Component Analysis (PCA) [17] is used to compress each block-level feature separately. The number of principal components that is suitable to compress a single feature (LFP, SP, DSP, VDSP, SCP, CP) depends on the underlying data and will always be a trade-off between quality and compression rate. In Section 5.3 we will therefore perform a set of tag classification experiments to identify an optimal trade-off between classification quality and compression rate. The next subsection briefly introduces the standard audio features that the presented block-level features are compared to in the evaluation.

3.4 'Standard' Audio Features

We compare the described block-level feature to a standard feature set that can easily and efficiently be extracted by the MARSYAS framework. The features are the **Spectral Centroid**, the **Rolloff**, the **Flux** and the **Mel-Frequency Cepstral Coefficients** (MFCC). Altogether, 16 numbers are extracted per audio frame. To capture some temporal information a running mean and standard deviation over a texture window of M frames is computed. The result intermediate features of the running mean computation still have the same rate as the original feature vectors. To come up with a single feature vector per song the intermediate running mean and standard deviation features are once more summarized by computing mean and standard deviation thereof. The overall result is a single 64-dimensional feature vector per audio clip. A more detailed description can be found in [18]. Finally, it is worth mentioning that all dimensions of both feature sets are always Min-Max normalized before they are used as input to the classification approach, which will be presented in the next section.

4. CLASSIFICATION APPROACH

As already mentioned we will use the classification method implemented in the MARSYAS framework to generate the tag predictions, for both features sets. MARSYAS implements a two stage classification schema (see figure 4) called "stacked generalization"[3]. In the first stage one audio feature vector per song serves as the input to a set of binary classifiers, one for each tag. In our case the binary classifiers are linear Support Vector Machines (SVMs) with probabilistic outputs. The probabilistic outputs of all binary classifiers of the first stage form the *tag affinity vector* (TAV). The TAV can be directly used to generate tag classifications by mapping the result for each tag either to 1 (tag present) or 0 (tag not present); the resulting vector is called *tag binary vector* (TBV). In MARSYAS this is realized via a thresholding approach. The threshold for each tag is chosen such that the number of testing songs associated with a given tag is proportional to the frequency of the tag in the training set. In our evaluation we will denote the results obtained via this first classification stage *stage 1 results (S1)*.

However, instead of just binarizing the obtained tag affinity vector one can additionally make use of prior knowledge about tag co-occurrences by feeding the obtained TAV into a second classification stage, which consists once again of one linear Support Vector Machine classifier per tag, but now with the TAV as input. As in the first stage the probabilistic output of the second stage classifiers can be interpreted as tag affinity vector. To distinguish the probabilistic output of the second stage from the probabilistic output of the first stage, the former is called *stacked tag affinity vector* (STAV). The binary classification result, called *stacked tag binary vector* (STBV) or *stage 2 result (S2)*, is then obtained via the same thresholding approach as described for the first stage.

Although the stacked generalization approach as described clearly has some merits, there is one weak point

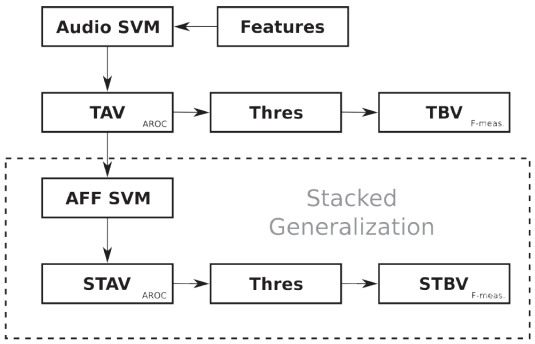


Figure 4. *Stacked Generalization classification schema, visualization from [3]*

in this schema, which is the specific thresholding strategy used by MARSYAS to generate the binary classification results (setting the threshold such that it leads to a certain percentage of positive predictions on the test set). It seems that this is an unconventional way of dealing with the class imbalance problem, which is one of the major problems in automatic tag classification. One future research direction will be to investigate more conventional approaches to deal with the class imbalance problem. In the following section we will present classification results for both stages (S1,S2) of stacked generalization tag classification.

5. EXPERIMENTS

In this section, we introduce the two datasets that are used in our evaluation, discuss the performance measures employed, and present the results of the experiments. We will first report on an evaluation of the applicability of PCA compression to the block-level feature set, and then present tag classification experiments for the direct comparison of the block-level and the standard audio feature set.

5.1 Datasets

5.1.1 CAL500

The CAL500 dataset [7] consists of 500 Western popular songs by 500 different artists. These songs have been annotated by 66 students with predefined semantic concepts that relate to six basic categories: instruments, vocal characteristic, genres, emotions, preferred listening scenario and acoustic qualities of a song (e.g. tempo, energy or sound quality). These concepts were then mapped to a set of 174 tags including positive and negative tags. Based on the user data binary annotation vectors were derived by ensuring a certain user agreement on the assigned tags. Figure 5 (left) shows the percentage of song that are annotated with each tag. Tags are sorted according to their annotation frequency. The most frequent tag in this dataset is applied to 88.4% of all songs. Typically, either a tag is used to annotate a majority of all songs or just for a few songs. Figure 5 (b) shows that the 90 most frequently applied tags account for 89.2% of all annotations.

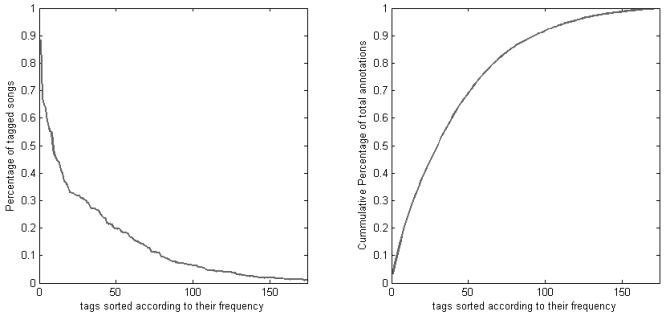


Figure 5. *Percentage of annotated songs per tag (left) and percentage of accumulated annotations of the first k most frequent tags (CAL500).*

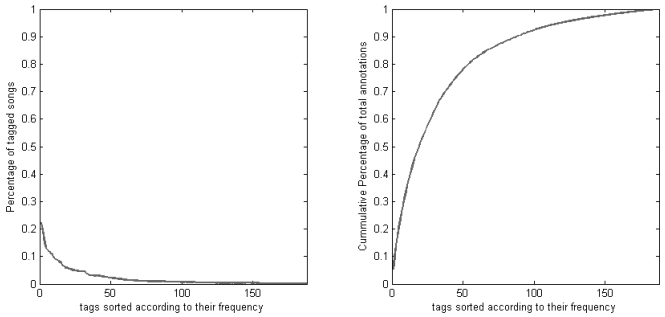


Figure 6. *Percentage of annotated songs per tag (left) and percentage of accumulated annotations of the first k most frequent tags (Magnatagatune).*

5.1.2 Magnatagatune

The second dataset in our evaluation is the Magnatagatune [19] dataset. This huge dataset contains 21642 songs annotated with 188 tags. The tags were collected by a music and sound annotation game, the TagATune [4] game. The dataset also contains 30 seconds audio excerpts of all songs that have been annotated by the players of the game. All the tags in the dataset have been verified (i.e. a tag is associated with an audio clip only if it is generated independently by more than 2 players, and only tags that are associated with more than 50 songs are included). From the tag distribution (figure 6) one can see that in terms of binary decisions (tag present / not present), the classification tasks are even more skewed than on the CAL500 dataset. The most frequently used tag applies to 22.42% of all songs only. 110 out of the 188 tags are used for less than 1% of all songs. From figure 6 one can see that the 87 most frequently used tags account for 89.86% of all annotations. This dataset is rather difficult to handle because of its size and the extremely skewed class distributions. To our knowledge, only Ness et al. [3] have so far presented results for this dataset.

5.2 Performance Measures

As automatic tag classification is a relatively new research area the performance measures used for evaluation vary significantly. Accuracy, precision, recall, f-score, true positive rate and true negative rate have been used. The only measure that is used in all evaluations is the f-score. Therefore, we will use the f-Score (see Eq.5 below) as one performance measure. As a second quality measure we use the G-mean (8) [20], which is a combination of Sensitivity (Acc^+), also known as true positive rate, and Specificity (Acc^-), also known as true negative rate. As such, it is a nice and compact measure that has the advantage of taking the class imbalance into consideration. We believe that these two quality measures together yield a compact and also comprehensive evaluation. Both measures can be computed globally over the entire (global) binary tag classification matrix, or separately for each tag and then averaged across tags. To differentiate between global and averaged performance measures the averaged per tag measures are named *avg. F-Score* and *avg. Gmean*, respectively. It is also important to note that we focus on the specific subtask of tag classification in this paper and therefore do not report on performance measures related to tag probabilities (or tag affinities) like average AUC-ROC.

$$\text{f-Score} = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision} + \text{recall})} \tag{5}$$

$$\text{Acc}^- = \text{TN}/(\text{TN} + \text{FP}) \tag{6}$$

$$\text{Acc}^+ = \text{TP}/(\text{TP} + \text{FN}) \tag{7}$$

$$\text{G-mean} = (\text{Acc}^- \times \text{Acc}^+)^{\frac{1}{2}} \tag{8}$$

5.3 Evaluation of the PCA Compression

To make the block-level features applicable to the task of automatic tag classification, we have to reduce their dimensionality in order to make the classification computationally tractable. As already discussed above, we use a standard Principal Component Analysis (PCA) to achieve this. However, both the compression rate and the achievable classification quality clearly depend on the number of principal components used to represent each block level-feature. In our evaluation we determine the number of principal components individually for each pattern (LFP, SP, DSP, SCP, CP, VDSP), based on the total variance captured by the k most important principal components. For example, given that we want to keep 80% of the total variance we compute the PCA for each pattern and then keep the number of principal components such that at least 80% of the total variance is accounted for. Thus, the prespecified variance determines the resulting feature size for each pattern individually.

To evaluate the PCA compression for different percentages of the total variance the CAL 500 dataset and two fold cross-validation is used. To compute the principal components only the features of the training set were used to prevent possible overfitting effects. As a consequence the dimensionality of the compressed feature set differs for the two cross-validation folds. The same split into two cross-validation folds were used for all experiments. The evaluation results are summarized in table 1. One can see that a feature set capturing about 70% to 80% of the total variance seems optimal in terms of tag classification quality.

[4] http://www.tagatune.org

Interestingly, also even a extreme reduction of the feature space to only about 37 dimensions performs comparably well. With respect to the stage 1 predictions some PCA compressed feature sets even outperform the original feature set. Furthermore, the second classification stage yields an improvement over the first for all evaluated feature sets. The best classification performance, however, is achieved by the uncompressed feature set using stacked generalization. It is also important to note that the decay in classification quality with a high number of principal components is related to the low number of data points that are available for the projection: the CAL500 consists of only 500 songs, in a feature space with 9448 dimensions. Altogether, we can conclude from these experiments that the proposed PCA compression approach does not diminish the tag prediction quality too much and is therefore a reasonable approach to reduce the size of the feature space.

5.4 Evaluation of the Feature Set Comparison

To compare the two feature sets we report on the presented performance measures obtained via 2-fold cross-validation on two different datasets (CAL500 and Magnatagatune). The same cross-validation split was used for the evaluated feature sets. These results are summarized in table 2, which gives the global performance measures computed over the global binary classification matrix, and the averaged per-tag performance measures. SAF denotes the standard audio feature set and BLF-PCA denotes the PCA compressed block-level feature set. BLF-FULL denotes the result of the uncompressed block-level feature set, which we only report for the smaller CAL500 dataset, because it was computationally not tractable on the larger Magnatagatune dataset. On the CAL500 dataset the BLF-PCA feature set consists of the 75 most important principal components capturing 75% of the total variance. On the larger Magnatagtune dataset the same variance threshold of 75% was used. For each performance measure the highest score on each dataset is highlighted in bold face. Clearly, the BLF-PCA feature set outperforms the standard feature set (SAF). An interesting finding is that the uncompressed block-level feature set (BLF-FULL) performs poorly on the first classification stage and obtains the highest scores for the second classification stage. We speculate that the bad performance in the first classification stage is related to the high dimensionality of this feature set. Another interesting detail is that the achievable gain in quality due to the improved feature set is in many cases relatively bigger than the gain from the second classification stage. Altogether, we can conclude that independent of the overall performance measure, either global or averaged per tag, the compressed block-level feature set compares favorably to a standard feature set.

6. CONCLUSIONS AND FUTURE WORK

In this paper have compared a set of recently proposed block-level features to standard audio features with respect to the task of automatic tag classification. We have shown that the proposed block-level feature set compares favorably to a standard feature set for the evaluated tag classification approach on two different datasets. Since the evaluated system with the standard features took the second rank in the MIREX 2009 tag classification task, we can conclude that the same system with the block-level features instead of the standard features is a state-of-the-art tag classification system.

Future research directions will include the exploration of standard techniques to account for the class imbalance problem of tags (e.g., over- or under-sampling [20]), instead of the rather unconventional threshold approach. Another interesting research direction will be to follow the idea of West et al. [2] and try to use automatically estimated tag and genre affinities for music similarity estimation.

7. ACKNOWLEDGMENTS

This research was supported by the Austrian Research Fund (FWF) under grant L511-N15.

8. REFERENCES

[1] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. of the 25th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR-02)*, 2002.

[2] K. West, S. Cox, and P. Lamere, "Incorporating machine-learning into music similarity estimation," in *Proc. of the 1st ACM Workshop on Audio and Music Computing Multimedia (AMCMM-06)*, 2006.

[3] S. R. Ness, A. Theocharis, G. Tzanetakis, and L. G. Martins, "Improving automatic music tag annotation using stacked generalization of probabilistic svm outputs," in *Proc. of the 17th ACM Int. Conf. on Multimedia (MM -09)*, 2009.

[4] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere, "Autotagger: A model for predicting social tags from acoustic features on large music databases," *Journal of New Music Research*, 2008.

[5] K. Seyerlehner and M. Schedl, "Block-level audio feature for music genre classification," in *online Proc. of the 5th Annual Music Information Retrieval Evaluation eXchange (MIREX-09)*, 2009.

[6] K. Seyerlehner, G. Widmer, and T. Pohle, "Fusing block-level features for music similarity estimation," in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, 2010.

[7] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech, and Language Processing*, 2008.

[8] M. Hoffman, D. Blei, and P. Cook, "Easy as cba: A simple probabilistic model for tagging music," in *Proc. of the Int. Sym. on Music Information Retrieval*, 2009.

% of total variance	dimensions (fold 1 / fold 2)	f-Score (S1)	G-Mean (S1)	f-Score (S2)	G-Mean (S2)
0.65	37 / 36	0.4812	0.6612	0.4863	0.6651
0.70	49 / 47	0.4807	0.6609	0.4911	0.6687
0.75	66 / 64	0.4844	0.6637	0.4965	0.6727
0.80	89 / 88	0.4768	0.6579	0.4983	0.6740
0.85	126 / 124	0.4570	0.6429	0.4849	0.6715
0.90	190 / 189	0.4187	0.6131	0.4895	0.6675
0.95	321 / 316	0.3759	0.5785	0.4886	0.6668
0.99	655 / 647	0.3037	0.5162	0.4678	0.6511
uncompressed	9448	0.4201	0.6142	0.5015	0.6764

Table 1. *G-Mean and f-Score performance results of classification stage 1 (S1) and classification stage 2 (S2) for various compression levels (CAL500)*

feature set	dataset	f-Score (S1)	G-Mean (S1)	f-Score (S2)	G-Mean (S2)
SAF	CAL500	0.4373	0.6277	0.4582	0.64387
BLF-FULL	CAL500	0.4201	0.6142	**0.5015**	**0.67641**
BLF-PCA	CAL500	**0.4844**	**0.6637**	0.4965	0.6727
SAF	Magnatagatune	0.3775	0.6101	0.3962	0.6252
BLF-PCA	Magnatagatune	**0.4163**	**0.6410**	**0.4201**	**0.6439**
feature set	**dataset**	**avg. f-Score (S1)**	**avg. G-Mean (S1)**	**avg. f-Score (S2)**	**avg. G-Mean (S2)**
SAF	CAL500	0.2401	0.3584	0.2577	0.3851
BLF-FULL	CAL500	0.2601	0.3910	**0.3061**	**0.4454**
BLF-PCA	CAL500	**0.2863**	**0.4233**	0.2908	0.4217
SAF	Magnatagatune	0.1777	0.3573	0.1932	0.3784
BLF-PCA	Magnatagatune	**0.2136**	**0.4019**	**0.2185**	**0.4081**

Table 2. *Comparison of standard audio features (SAF) and block-level features (BLF) for stage 1 (S1) and stage 2 (S2) of the stacked generalization tag classification approach.*

[9] E. Pampalk, A. Rauber, and D. Merkl., "Content-based organization and visualization of music archives," in *Proc. of the 10th ACM Int. Conf. on Multimedia*, 2002.

[10] A. Rauber, E. Pampalk, and D. Merkl, *The SOM-enhanced JukeBox: Organization and visualization of music collections based on perceptual models.* Journal of New Music Research, 2003.

[11] T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification.," in *Proc. of the 6th Int. Conf. on Music Information Retrieval (ISMIR-05)*, 2005.

[12] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, 2007.

[13] M. Goto, "Smartmusickiosk: Music listening station with chorus-search function.," in *Proc. of the 16th ACM Symp. on User Interface Software and Technology (UIST-03)*, 2003.

[14] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, "On rhythm and general music similarity," in *Proc. of the 10th Int. Society for Music Information Retrieval Conf. (ISMIR-09)*, 2009.

[15] J. H. Jensen, M. G. Christensen, and S. Jensen, "A tempo-insensitve representation of rhythmic patterns," in *Proc. of the 17th European Signal Processing Conf. (EUSIPCO-09)*, 2009.

[16] D. Jiang, L. Lu, H. Zhang, J. Tao, and L. Cai, "Music type classification by spectral contrast feature," in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME-02)*, 2002.

[17] I. T. Jolliffe, *Principal Component Analysis*. Springer, second ed., October 2002.

[18] G. Tzanetakis and P. Cook, "Musical genre classification of audio signal," *IEEE Transactions on Audio and Speech Processing*, 2002.

[19] E. Law and L. von Ahn, "Input-agreement: A new mechanism for collecting data using human computation games," in *Proc. of the 27th Int. Conf. on Human Factors in Computing Systems (CHI-09)*, 2009.

[20] B. Wang and N. Japkowicz, "Boosting support vector machines for imbalanced data sets," *Knowledge and Information Systems*, 2009.

[21] G. Tzanetakis, "Marsyas submission to mirex 2009," in *Online Proc. of the 5th Annual Music Information Retrieval Evaluation eXchange (MIREX-09)*, 2009.

ADDITIONAL EVIDENCE THAT COMMON LOW-LEVEL FEATURES OF INDIVIDUAL AUDIO FRAMES ARE NOT REPRESENTATIVE OF MUSIC GENRE

Gonçalo Marques[1], **Miguel Lopes**[2], **Mohamed Sordo**[3], **Thibault Langlois**[4], **Fabien Gouyon**[2]
[1]DEETC - ISEL Lisboa, [2]INESC Porto, [3]Universitat Pompeu Fabra, Barcelona, [4]DI - FCUL Lisboa
[1]gmarques@isel.pt

ABSTRACT

The Bag-of-Frames (BoF) approach has been widely used in music genre classification. In this approach, music genres are represented by statistical models of low-level features computed on short frames (e.g. in the tenth of ms) of audio signal. In the design of such models, a common procedure in BoF approaches is to represent each music genre by sets of instances (i.e. frame-based feature vectors) inferred from training data. The common underlying assumption is that the majority of such instances do capture somehow the (musical) specificities of each genre, and that obtaining good classification performance is a matter of size of the training dataset, and fine-tuning feature extraction and learning algorithm parameters.

We report on extensive tests on two music databases that contradict this assumption. We show that there is little or no benefit in seeking a thorough representation of the feature vectors *for each class*. In particular, we show that genre classification performances are similar when representing music pieces from a number of different genres with the same set of symbols derived from a single genre or from all the genres. We conclude that our experiments provide additional evidence to the hypothesis that common low-level features of isolated audio frames are not representative of music genres.

1. INTRODUCTION

A large literature exists on automatic classification of music pieces based on raw audio data. Providing a complete review is out of the scope of this paper, interested readers are referred to [1] and [2]. Most approaches to date share the same underlying algorithmic architecture [1]: the Bag-of-Frame (BoF) approach. Music genres are represented via long-term statistical models of large collections of feature vectors computed on short frames of audio signal (on the scale of tenth of ms). In the BoF approach, it is implicit that all frames have a similar information load, and that all are significant in the modeling of genres. A prototypical implementation of this architecture, now considered standard procedure, uses Gaussian Mixture Modeling (GMMs) of short-term Mel-Frequency Cepstral Coefficients (MFCCs).

Aucouturier [1] discusses the existence of a "glass-ceiling" to the performance of the BoF approach to music genre classification. He argues that it is fundamentally bound and that a change of paradigm to music genre modeling is needed. A number of recent papers also challenge the BoF approach arguing that all frames may not have the same information load and propose to train models of music genre on a selection of the available training data, either the most representative, or the most discriminative [3, 4, 5]. In previous research on the topic of instance selection for music genre classification [6], we showed that when representing music signals by common low-level features (i.e. MFCCs and spectral features), similar classification accuracies could be obtained when training classifiers on all of the training data available, or on few training data instances from each class.

In this paper, we go a step further and propose the hypothesis that values of common low-level features on isolated signal frames are *not* representative of music genres. In other words, the performance of BoF music genre models may be bound because there would be not such thing as a "typical e.g. Rock, or Classical frame" (more precisely, no such thing as "typical sets of low-level audio feature values for a e.g. Rock, or Classical frame").

To address this hypothesis, we conduct systematic experiments in which models of music genres are built with training data representative of *only part of the genres* from the dataset.

These experiments imply (1) the definition of a codebook, generated from different partitions of some available training data (section 2), and (2) the expression of training examples from each genre with this codebook and the actual training of genre models (section 3.1). In the remainder of section 3, we describe experimental details regarding data and audio features used. Section 4 reports results of our experiments and in section 5, we propose some conclusions and directions for future research.

2. CODEBOOK GENERATION PROCEDURES

Following the technique described in [5], the experiments reported in this paper are based on a codebook approach. The feature vectors extracted from the training set are processed in order to select a limited number of representative feature vectors that constitute the codebook. We ex-

perimented several approaches for the constitution of the codebook, including selecting the centroids obtained with k-means, selecting the most representative feature vectors according to the probability density function modeled with a Gaussian Mixture Model, and combinations of both approaches (see [5] for more details). In this paper, to avoid any particular bias, we use random selection of feature vectors, as follows.

Codebooks are generated by randomly selecting k_1 feature vectors from each music piece and then selecting k_2 feature vectors in the set of $N \times k_1$ feature vectors (where N corresponds to the number of music pieces in the training set). In both cases a uniform distribution is used. For all experiments described in this paper we used $k_1 = 20$ and $k_2 = 200$.

Notice that the creation of codebooks is an unsupervised process, i.e. each music piece is processed independently of the class it belongs to. Three kinds of codebooks were generated:

Using data from all genres but one The codebook is generated ignoring class X. This is repeated for each class. The codebooks obtained this way are called "all-but-X".

Using data from a single genre In this case codebooks are generated using the feature vectors found in music pieces from only one genre. These codebooks are referred as "only-X".

Using data from all genres As a base for comparison, we generated codebooks that use the data from all classes, as described previously. These codebooks are called "all-genres".

3. EXPERIMENTS

3.1 Classification models

3.1.1 Data representation by vector quantization

Input data to the classifiers (both for training and testing) is based on a codebook approach: each music piece is first converted into a sequence of discrete symbols pertaining to one of the codebook symbols considered here, through vector quantization of the audio features. More precisely, for each music piece, the feature vector of each frame is assigned a symbol corresponding to the nearest symbol in the set of $k_2 = 200$ possible symbols of the given codebook (see section 2).

Finally, depending on the classifier, each music piece is represented by either a normalized histogram of the symbols frequency, or the sequence of symbols itself.

3.1.2 Histogram + k-NN

The k-NN algorithm treats the histograms as points in a k_2 dimensional space. The music pieces in the training set are used as examples, and a new music piece is classified by a majority vote of its neighbors. In our experiments, we used a 5-NN classifier. The nearest neighbors were calculated based on two distance metrics: the Euclidean distance, and a symmetric version of the Kullback-Leibler divergence:

$$\mathcal{D}\left(P\|Q\right) = D_{KL}\left(P\|Q\right) + D_{KL}\left(Q\|P\right) \qquad (1)$$

where, Q and P are the normalized histograms of two music pieces, and

$$D_{KL}\left(P\|Q\right) = \sum_{i=1}^{k_2} P(i) \log \frac{P(i)}{Q(i)} \qquad (2)$$

is the Kullback-Leibler divergence, and $P(i)$ is the i-bin of the histogram P. In order to use this divergence, the distributions P and Q must have non-zero entries. However, this can happen if one or more symbols from the codebook are not used in the representation of a given music piece. To overcome this limitation, we add one hit to all histogram bins before performing the histogram normalization.

3.1.3 Histogram + SVM

A Support Vector Machine (SVM) [7] was used with a Radial Basis Function kernel with $\gamma = 1/k_2$ (where k_2 is the number of features, i.e. 200), and a cost $C = 2000$. Experiments for determining optimal parameter values are left for future work.

3.1.4 Markov models

This classification method is based on Markov modeling. A Markov model is build for each genre, by computing the transition probabilities (bigrams) for each group of sequences [5]. The outcome is set of transition matrices, one for each class, containing the probabilities, $P(s_j|s_i)$, of each symbol s_j given the preceding symbol s_i. For classification, the test music piece is converted into a sequence of symbols, $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, and the (logarithmic) probability of the sequence given each model is calculated:

$$\begin{aligned} \mathcal{L}_M(\mathcal{S}) &= \log\left(P_M(s_{i=1,\ldots,n})\right) \\ &= \log\left(P_M(s_1)\right) + \sum_{i=2}^{n} \log\left(P_M(s_i|s_{i-1})\right) \end{aligned} \qquad (3)$$

where P_M represents the symbols probability mass function for the model M. The music class is chosen by the model with the highest score $\mathcal{L}_M$.

3.2 Data

Two datasets were used in our experiments. The first one is a subset of the Latin Music Database (henceforth, "LMD dataset"), and the second is the ISMIR 2004 Genre Classification Contest (henceforth, "ISMIR04 dataset").

3.2.1 LMD

The Latin Music Database [8] is composed of 3,227 full-length music pieces, uniformly distributed over 10 genres: Axé, Bachata, Bolero, Forró, Gaúcha, Merengue, Pagode, Salsa, Sertaneja, and Tango. For our experiments, we created a subset of 900 music pieces, which are divided in three folds of equal size (30 pieces per class). We used an artist filter [9, 10] to ensure that the music pieces from a

specific artist are present in one and only one of the three folds. We also added the constraint of the same number of artists per fold.

3.2.2 ISMIR04

This dataset was created for the genre classification contest organized during the ISMIR 2004 conference [11], [1] and is divided in six genres with a total of 729 music pieces for training and 729 music pieces for testing. The music piece distribution among the six genres is: 320 Classical, 115 Electronic, 26 JazzBlues, 45 MetalPunk, 101 Rock-Pop, and 122 World. As in the original ISMIR 2004 contest, the dataset does not account for artist filtering between both sets.

3.3 Audio Features

We used the free MARSYAS framework [2] to extract 17 audio features from 46ms frames of the audio signals (mono, sampled at 22050Hz, no overlap). The features are commonly used in audio genre classification tasks: the zero crossing rate, spectral centroid, rolloff frequency, spectral flux, and 13 MFCCs, including MFCC0.

3.4 Evaluation Metrics

We report the accuracy obtained over test sets only, both for the ISMIR04 and LMD datasets.

On the evaluation on the ISMIR04 dataset, we kept the original training-testing division proposed in the ISMIR 2004 genre classification contest.

The evaluation on the LMD dataset follows a three-fold cross validation procedure: two folds are used for training and one for testing, with all the permutations of the folds. The performance measure is the accuracy averaged over the three test runs.

4. RESULTS AND DISCUSSION

First of all, it is interesting to notice that results obtained on the ISMIR04 and LMD datasets are comparable to state-of-the-art results. For instance, the best result obtained on ISMIR04 is 79.8%, which is very similar to the results obtained by the best algorithms in the last MIREX in which this data was used (i.e. MIREX 2005). [3] The best result obtained on LMD is 64.9%, when the best result on this dataset in MIREX 2009 was 74.6% and the average accuracy accounting for all participants was 55.5%.

In almost every set of experiment we found that the classifiers based on Markov models is better than the three other alternatives. This observation tends to confirm the fact that the information contained in the temporal sequence is indeed relevant to the classification into genres.

Tables 1 and 2 show the overall classification accuracy for the ISMIR04 and the LMD datasets respectively when one genre is not used in the codebook generation process. The lines represent accuracy scores obtained with different classification procedures (columns). For the sake of comparison, the first line contains the results obtained with a codebook computed with all genres.

ISMIR04 — all-but-one genre				
	Markov	SVM	k-NN	k-NN$_{\mathrm{KL}}$
all genres	79.0	**68.6**	73.0	**76.0**
all-but-Class.	**79.3**	68.3	70.8	73.7
all-but-Elec.	**79.7**	**68.6**	69.3	73.7
all-but-JaBl.	78.3	67.8	70.6	74.6
all-but-MePu.	**79.3**	68.5	71.5	75.9
all-but-RoPo.	78.6	68.2	**73.3**	75.3
all-but-Wor.	**79.1**	68.2	**73.3**	74.3
Average	79.1	68.3	71.5	74.6

Table 1. Results for the ISMIR04 dataset. Each line represents the average accuracy (over all genres) obtained with codebooks generated from all but one genre. The last line contains the average of lines 2 to 7. The first line contains the results obtained with a codebook computed with all genres. Results in bold are those that outperform those of the first line.

From these experiments (Tables 1 and 2) one can see that when the feature vectors from one class are ignored in the creation of the codebook, we do not observe a severe decrease of the accuracy. In some cases the accuracy obtained without one of the classes is equal or better than when all genres are used (numbers with bold font).

LMD — all-but-one genre				
	Markov	SVM	k-NN	k-NN$_{\mathrm{KL}}$
all genres	64.0	54.2	47.0	52.2
all-but-Axé	62.7	**56.6**	**50.1**	**52.7**
all-but-Bach.	**64.9**	54.1	**48.1**	**52.7**
all-but-Bole.	63.0	**54.9**	**49.9**	**53.0**
all-but-Forr.	61.1	53.8	**48.9**	51.2
all-but-Gáu.	63.6	53.1	**48.3**	50.9
all-but-Mer.	63.7	53.6	**47.8**	50.4
all-but-Pag.	63.5	54.0	**49.1**	**52.7**
all-but-Sals.	63.5	53.7	**47.9**	50.8
all-but-Sert.	62.8	**54.3**	**48.9**	51.4
all-but-Tan.	63.1	**54.3**	**48.9**	51.9
Average	63.2	54.2	48.8	51.8

Table 2. Results for the LMD dataset. Each line represents the average accuracy (over all genres) obtained with codebooks generated from all but one genre. The last line contains the average of lines 2 to 11. The first line contains the results obtained with a codebook computed with all genres.

Tables 3 and 4 are very similar to table 1 and 2, but in this case, the codebooks were computed using feature vectors from only one genre.

It can be seen that reducing dramatically the universe of feature vectors, the average accuracy compared to the case where all genres are used is not substantially different.

[1] http://ismir2004.ismir.net/ISMIR_Contest.html
[2] http://marsyas.sness.net/
[3] http://www.music-ir.org/mirex/2005/

ISMIR04 — only one genre				
	Markov	SVM	k-NN	k-NN$_{\mathrm{KL}}$
all genres	79.0	**68.6**	**73.0**	**76.0**
only-Class.	76.3	62.8	66.8	71.7
only-Elec.	**79.4**	67.2	70.9	73.4
only-JaBl	78.5	66.8	67.2	72.7
only-MePu.	74.8	66.9	65.3	72.3
only-RoPo.	75.7	67.2	70.5	75.4
only-Wor.	**79.8**	67.5	69.8	73.3
Average	77.4	66.4	68.4	73.1
All − Average	1.6	2.2	4.6	2.9

Table 3. Results for the ISMIR04 dataset. Each line represents the accuracy obtained with codebooks generated with data from a single genre. The last line shows the decrease in accuracy between results obtained with a codebook generated with data from all genres and average results obtained with a codebook generated with data from a single genre.

LMD — only one genre				
	Markov	SVM	k-NN	k-NN$_{\mathrm{KL}}$
all	64.0	54.2	47.0	52.2
only-Axé	59.9	**54.4**	**49.6**	**52.2**
only-Bach.	**64.9**	53.7	45.8	51.7
only-Bole.	62.1	53.0	45.6	49.0
only-Forr.	**64.7**	53.9	**49.6**	**52.9**
only-Gáu.	**64.6**	**54.6**	**49.8**	**55.8**
only-Mer.	62.3	53.6	**48.7**	51.0
only-Pag.	63.5	53.0	**48.5**	50.8
only-Sals.	63.9	53.1	**47.2**	50.9
only-Sert.	61.9	53.3	**49.2**	**54.3**
only-Tan.	55.0	46.2	40.1	43.6
Average	62.3	52.9	47.4	51.2

Table 4. Results for the LMD dataset. Each line represents average results obtained with codebooks generated with data from a single genre.

In the case of the ISMIR04 dataset, using only one genre for building the codebook leads to an average decrease of 1.6 percentage points for Markov models, 2.2 percentage points for SVM, 4.6 percentage points for k-NN and 2.9 percentage points for k-NN$_{\mathrm{KL}}$. It is interesting to note that the non-parametric method (k-NN) is the most affected by a reduction of the amount of data. However, we can also see that, at least for the Markov model classifier, in some cases performance can be better when using only one genre to build the codebook.

In the case of the LMD dataset (Table 4), we observe that, in numerous cases, the accuracy obtained with codebooks modeled after only one genre is equal or better than the one obtained using all genres. From these experiments we can see that using a small subset of the feature vectors, even if they belong to only one genre, we are still able to build a classifier that performs well.

ISMIR04				
	all	all-but-1	only-1	Diff.
Classical	**96.9**	95.3	96.6	1.3
Electronic	71.1	72.8	**73.7**	0.9
JazzBlues	63.4	69.3	**76.9**	7.6
MetalPunk	71.1	**75.6**	64.4	−11.2
RockPop	61.8	61.8	**64.7**	2.9
World	59.9	60.7	**63.1**	2.4

Table 5. Breakdown with respect to genres of the results for the ISMIR04 dataset, using Markov models classifiers. Each row shows the accuracy observed for the corresponding class with the three different kinds of codebooks. For instance, the entry in the fourth line second column (75.6) is the percentage of correctly classified music pieces for the class MetalPunk, using a codebook computed with feature vectors from all genres but MetalPunk. The last column contains the difference between the only-1 and all-but-1 accuracies.

LMD				
	all	all-but-1	only-1	Diff.
Axé	**44.4**	41.1	36.7	−4.4
Bach.	83.3	84.4	**86.7**	2.3
Bole.	76.7	74.4	**82.2**	7.8
Forr.	**37.8**	35.6	35.6	0.0
Gaúc.	44.4	47.8	**53.3**	5.5
Mere.	80.0	76.7	**81.1**	4.4
Pago.	56.7	56.7	**60.0**	3.3
Sals.	68.9	65.6	**72.2**	6.6
Sert.	61.1	54.4	**64.4**	10.0
Tang.	**86.7**	85.6	85.6	0.0

Table 6. Breakdown with respect to genres of the results for the LMD dataset, using Markov models classifiers. Each row show the accuracy observed for the corresponding class with the three different kinds of codebooks. The last column contains the difference between the only-1 and all-but-1 accuracies.

Since the performance is measured on all classes, a lower classification rate on one class may be hidden by higher scores on others. Therefore we evaluated the accuracy obtained for each class with each of the three ways of building the codebooks. These results are shown in tables 5 and 6. In the case of the ISMIR04 dataset (table 5), one can see that the differences in accuracy between using only a given class (4th column) and not using that class at all in the generation of the codebook (3rd column) is rather small with two exceptions: JazzBlues and MetalPunk, albeit in an opposite way. These exceptions may be explained by the fact that both categories are represented with a very small number of music pieces (26 for JazzBlues and 45 for MetalPunk).

We also studied the effect of using only one class for codebook creation on the accuracy observed on other classes. The results are shown in table 7 for the ISMIR04 dataset

ISMIR04							
	only-Class.	only-Elec.	only-JaBu	only-MePu.	only-RoPo.	only-Wor.	Diff.
Class.	**96.6**	95.9	95.6	91.3	91.6	95.9	0.7
Elec.	69.3	**73.7**	71.1	67.5	71.1	72.8	0.9
JaBu.	57.7	69.2	**76.9**	61.5	65.4	65.4	7.7
MePu.	77.8	73.3	77.8	64.4	73.3	**82.2**	4.4
RoPo.	51.0	61.8	59.8	63.7	**64.7**	59.8	1.0
Wor.	54.1	60.7	56.6	54.1	50.8	**63.1**	2.4

Table 7. Genre breakdown results for the ISMIR04 dataset using Markov models with different codebooks based on only one class. The table entries are class accuracies (lines) for a given codebook (columns). The last column shows the difference between the best (bold) and the second best accuracy (underlined) of each row.

LMD											
	only-Ax.	only-Ba.	only-Bo.	only-Fo.	only-Gá.	only-Me.	only-Pa.	only-Sa.	only-Se.	only-Ta.	Diff.
Axé	36.5	42.2	22.1	44.4	45.6	**46.7**	37.8	44.4	45.6	37.8	1.1
Bach.	81.1	**86.7**	83.3	83.3	84.4	83.3	85.6	84.4	81.1	83.2	1.1
Bole.	76.7	77.8	**82.2**	68.9	74.4	70.0	70.0	72.2	68.9	80.0	2.2
Forr.	34.4	34.4	32.2	35.6	32.2	34.4	**41.1**	33.3	32.2	36.7	4.4
Gáu.	44.4	51.1	43.3	**53.3**	**53.3**	48.9	44.4	51.1	45.6	33.3	2.2
Mer.	75.6	76.7	77.8	80.0	77.8	**81.1**	75.6	77.8	76.7	72.2	1.1
Pag.	50.0	57.8	55.6	**60.0**	**60.0**	48.9	**60 .0**	57.8	53.3	50.0	2.2
Sals.	61.1	**74.4**	72.2	72.2	72.2	63.3	61.1	40.3	65.6	42.2	2.2
Sert.	52.2	61.1	55.6	**64.4**	58.9	62.2	58.9	60.0	**64.4**	28.9	2.2
Tan	86.7	86.7	84.4	84.4	87.8	84.4	**88.9**	85.6	85.6	85.6	1.1

Table 8. Genre breakdown results for the LMD dataset using Markov models with different codebooks based on with only one class. The table entries are class accuracies (lines) for a given codebook (columns). The last column shows the difference between the best (bold) and the second best accuracy (underlined) of each row.

and in table 8 for the LMD dataset. Each row of these tables contain the accuracy observed on one class (rows) when using a codebook based on each single class (columns). Values in bold font correspond to the maximum of each row and can be interpreted as the best codebook for the representation of each class. For example, in table 7 we can see that all classes but MetalPunk are better represented by a codebook defined using the same class. But if we look at the second best accuracy (underlined numbers) we can see that using feature vectors from a different class can lead to seemingly similar performance. The difference between best and second best accuracy is shown in the last column. For example, Classical music may be represented by feature vectors that belong to Electronic or World music losing only 0.7 percentage points in accuracy. RockPop may be represented by MetalPunk or Electronic feature vectors, losing only 1 percentage point. Counter examples can be found with the cases of JazzBlues and MetalPunk but this may be caused by the fact that those classes are represented by a small number of music pieces when compared to other classes. It is notable that in some cases (such as Classical and Electronic) the ability of using a genre to represent another genre occurs with genres that are perceived very differently by listeners.

When looking at table 8, which describes the same experiments with the LMD dataset, one can see that the difference between the best and the second best accuracy is small, showing that a genre may be represented using feature vectors from another genre without losing too much accuracy, and in some cases even increasing accuracy apparently.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we tackle the problem of music genre classification with low-level features computed on collections of audio frames. In the common approach to this problem, it is generally assumed that the majority of frames of a particular genre (or, more precisely, their representations via MFCCs and other common low-level features) carry information that is specific to that genre. The main conclusion of our experiments is that common low-level features computed on individual audio frames are in fact not representative of music genres (even if their distributions are). We demonstrate that seeking the most extensive and thorough representation of each genre with respect to such low-level features does in fact not bring any benefit in terms of classification accuracy.

Specifically, in our experiments, music pieces from diverse genres are represented by sequences of symbols from a codebook. This codebook is made up of feature vectors from either one, all, or a selection of genres. We show that the classification accuracy averaged over all genres is very similar when the codebook is derived from data of all

genres vs. data of all genres but one (tables 1 and 2), or vs. data of only one single genre (tables 3 and 4). This appears to be true for diverse classifiers. Further, the provenance of the data used for deriving the codebook does not seem to have a profound impact on classification accuracy of each particular genre (tables 5 and 6), even in the case where the data used comes from one single, different genre (tables 7 and 8). These results appear to hold for two different datasets of very different natures.

This is not to say that such frame-based representations are not useful for music genre classification, as they indeed permit to classify better than random. However, even if *collections* of frames can represent music genres with some success, we show here that *individual* frames do not.

Given the relatively small variations in accuracies for a given genre, and the fact that these variations go both ways (small decrease in some cases and small increase in some others), we suspect that statistical significance tests would show the near equivalence of accuracies over each genres. This is an avenue for future work.

We believe that the results detailed in this paper contribute to the emerging idea that significant improvements in music genre classification will require the design of better initial signal representations, features that carry information that would be specific to genres, closer to musical concepts [12].

6. ACKNOWLEDGMENTS

Thanks to Jaime Cardoso at FEUP/INESC Porto, Alessandro L. Koerich at PPGIA-PUCPR and Luiz E. S. Oliveira at UFPR Curitiba, and to anonymous reviewers. This research was supported by Convénio FCT/CAPES 2009; *Fundação para a Ciência e a Tecnologia* (FCT) and QREN-AdI grant for the project Palco3.0/3121 in Portugal; *Ministerio de Educación* in Spain. This work was partially supported by FCT through LASIGE Multiannual Funding and VIRUS research project (PTDC/EIA-EIA/101012/2008). The first author is supported by PROTEC grant SFRH/PROTEC/50118/2009.

7. REFERENCES

[1] J.-J. Aucouturier, *Dix expériences sur la modélisation du timbre polyphonique*. PhD thesis, University Paris VI, 2006.

[2] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, 2006.

[3] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio patter recognition: A sufficient model for urban soundscapes but not for polyphonic music," *Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[4] U. Bagci and E. Erzin, "Automatic classification of musical genres using inter-genre similarity," *IEEE Signal Processing Letters*, vol. 14, no. 8, pp. 521–524, 2007.

[5] T. Langlois and G. Marques, "Music classification method based on timbral features," in *International Conference on Music Information Retrieval*, 2009.

[6] M. Lopes, F. Gouyon, C. Silla, and L. E. S. Oliveira, "Selection of training instances for music genre classification," in *International Conference on Pattern Recognition*, 2010.

[7] Y. EL-Manzalawy and V. Honavar, *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`.

[8] C. Silla, A. Koerich, and C. Kaestner, "The latin music database," in *International Conference on Music Information Retrieval*, 2008.

[9] E. Pampalk, *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Austria, 2006.

[10] A. Flexer, "A closer look on artist filters for musical genre classification," in *International Conference on Music Information Retrieval*, 2007.

[11] P. Cano, E. Gomez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack, "Ismir 2004 audio description contest," in *MTG Technical Report MTG-TR-2006-02*, 2006.

[12] J.-J. Aucouturier, "Sounds like teen spirit: Computational insights into the grounding of everyday musical terms," in *Language, Evolution and the Brain, Frontiers in Linguistics Series* (J. Minett and W. Wang, eds.), Taipei: Academia Sinica Press, 2009.

DYNAMIC CUES FOR NETWORK MUSIC INTERACTIONS

Alain B. Renaud
Bournemouth University, England
arenaud@bournemouth.ac.uk

ABSTRACT

This paper provides an overview of a cueing system, the Master Cue Generator (MCG) used to trigger performers (humans or computers) over an IP-based network. The performers are scattered in several locations and receive cues to help them interact musically over the network. The paper proposes a classification of cues that dynamically evolve and reshape as the performance takes place. This begets the exploration of various issues such as how to represent and port a hierarchy of control over a networked music performance (NMP) and also takes into account parameters inherent to a network such as latency and distance. This approach is based on several years of practice-led research in the field of NMP, a discipline that is gaining grounds within the music technology community both as a practice and through the development of tools and strategies for interacting over disparate locations.

1. INTRODUCTION

Performing in real time over high-speed networks is a now well-accepted paradigm and has become an integral part of Telematic Art, considered by authors such as Roy Ascott, *"as an artistic medium in itself"* [1]. There have been several attempts to achieve interactive telematic performances. However the first telematic concert, using high-speed research networks with no audio compression and thus allowing CD like audio quality took place between two spaces at Stanford University in 2000, with an ensemble split and performing about one kilometer apart [3]. This initial test was led by the Sound Waves over the Internet from Real-time Echoes (SoundWIRE) project founded in 1998 at Stanford University [3]. SoundWIRE, through various experiments and studies such as the "clapping experiment" [4], which measured a threshold in milliseconds for ensemble accuracy, set the grounds for further development in the field of networked music performance (NMP). The discipline of NMP branched out in many directions, and due to its nature, which involves being distributed, led to the involvement of several new participants. The work of SoundWIRE, however, demonstrated that it was possible to interact musically over a long distance despite the inherent latency of the network. The excitement of being able to play apart led to the challenge of choosing whether music performed over a network could be simply improvised or formally structured through the help of network-centric cueing mechanisms.

2. IMPROVISATION

The network provides a platform for sharing synchronization information and cues as it allows several performers to share a common infrastructure for exchanging common musical structures. Performing over the network introduces the principle of dislocation of performers as they are not in the same space but are playing in real time together whilst being located in several spaces. Free improvisation has been a practice often employed in NMP due to its emphasis on musician-to-musician interaction and flexibility of materials; thus providing a good basis for developing musical strategies for interacting over a network regardless of its latency. Free improvisation as outlined by Derek Bailey, *"pre-dates any other music – mankind's first musical performance could not have been anything other than a free improvisation*" [2]. It is therefore not surprising that new media environments, such as NMPs, resort to basic sorts of

musical forms, which do not involve a formal structure. NMP is such a recent practice that most performances will start from an empty shell, where the infrastructure will first be put into place followed by numerous tests to make sure that the communication works and finalised by a short rehearsal. The fact that the IP – based network is the medium that interconnects them will play a crucial role in the development of those social interactions through space and time. In this context, and based on several years of research in the field through large scale NMPs such as the Disparate Bodies series [7] as well as the experimentations of the Net. Vs. Net Collective [8], the development of coherent network centric cueing strategies was needed.

It is due to the fact that, very quickly the improvized performance requires some sort of formalization so that performers can be cued over the network, leading to the inclusion of a basic structure within the improvisation. In this context and as a result of the practice in the field, a formal classification of networked cues and how they can be used to interact musically over the network made sense.

3. CUES

3.1 Rationale

In order to provide an easy way to represent various cue information over the network, an integrated cueing system called the Master Cue Generator (MCG) was developed. The MCG aims to provide a rough standard to distribute cues over the network. The MCG has been used and tested in several NMPs such as the Disparate Bodies series [7] and with the Net. Vs. Net Collective [8]. The system is continuously being developed further with the goal to achieve a common cueing structured language for networked music improvisation.

The MCG was built with Max/Msp [6] and is able to send cues to a multitude of locations as standard OpenSoundControl (OSC) [9] messages, meaning that any OSC compliant application is able to receive the cues and converse back to the MCG should a direct feedback be necessary. The MCG was initially designed to function based on a client/server architecture. However, it was later discovered that modifications of the network configuration should be possible based on the changing attribution of roles, defined as who plays the role of the MCG in the network. This complex and challenging aspect is currently being developed.

Currently, the MCG broadcasts important musical information by providing a basic structure to the nodes playing over the network, such as which section of the piece the nodes are in, as well as warning messages that the piece is about to switch to another section. The types of cues and their specific nature can be customized depending on the artistic approach given to the piece.

Figure 1. The MCG engine

3.2 Types of Cues

There are three types of cues that have been so far identified as part of the classification: temporal behavioural and notational. All the cues below have been developed based on the practice in the field and the classification is constantly being updated as the practice progresses.

3.2.1 Temporal Cues

Temporal cues are sent out as information from the server to the nodes and are related to timing. Examples include the length of a cue, a warning that the cue is about to finish, or how much time a given node is in control of the improvisation until the given node delegates its control to another node and thus conceptually modifying the topology of the network. They are the most important types of cues, in order to keep the ensemble together and, thereby, can be synchronized with various audio triggering cues, which are indicators that the structured improvisation is about to change from one section to another.

3.2.2 Behavioral Cues

Behavioral cues are cues that are sent with a certain scenario attached to them. This can, for example, include the triggering of a waveform, or the suggestion that a given node needs to

play certain nodes only above the note C4. Behavioural cues can also trigger physical elements in a remote space such as the ringing of a distant bell or the triggering of an analogue synthesizer. Behavioural cues are more complex types of cues. They allow the broadcast of messages that will have an influence on the actual audio content of the piece being performed over the network. Behavioural cues are often part of the process of a structured network improvisation. An example is the triggering of pre-recorded waveforms that reside on remote computers.

The waveforms are being played back remotely but triggered by the MCG. Behavioural cues also have the potential to influence the actual frequency content of each remote node by broadcasting messages that will interpolate or cut-off. The MCG, in this case, provides the intelligence behind the system by ensuring that each node has a different frequency bandwidth. The frequency dependent interconnections that are being created in this case, also allow for a morphing of frequency range distributions across the nodes. For example, a node can decide to borrow a frequency range from another node, in which case the frequency distribution is swapped between nodes. Amplitude control is also an important type of behavioral cue. As part of the pre-defined structured improvisation, a distribution of amplitudes across the nodes can be implemented in advance.

It allows the distribution of intensities across the network and is a very democratically aware way of ensuring that each node can be properly identified during a performance. For example, in the case of a performance between three nodes, the MCG will make sure that in Section 1 of a piece, Node 1 is the loudest, while Node 2 is the quietest and Node 3 is at mid-level. As a result, reasonably complex interdependencies can be achieved by swapping loudness information between nodes as well as interpolating them. They are, of course, many other interaction cues that can be created and their types and resulting actions wholly depend on desired objectives in the design of the performance.

3.2.3 Notational Cues

Notational cues are able to display content that can be identified by the performers as being helpful in the good running of the performance. This can include the visualisation of the waveforms from each site, the display of the cue number, a countdown or dynamic shapes that can be activated by various factors in the performance. Transmitting concrete or abstract notation based information is a real challenge over a network due to the latency and of the different distances between the MCG and the nodes. Even though the MCG is capable of re-triggering events so that a cue information arrives simultaneously at all nodes, which is a punctual or periodical type of information, the triggering mechanism does not work well for continuous information and, thereby, a drift is likely to occur over a period of time. Therefore, notational cues have traditionally been sent over the network in a punctual fashion, where the graphical representation is analogous to a slide show. If some events are of continuous nature, they tend to be transferred to remote nodes before the performance and triggered remotely. In order to efficiently represent the cues and the synchronicity information, a set of visualisation tools has been developed to simply, but efficiently, display score information on various sites.

3.2.4 Active/Passive Cues

The three types of cues identified above can have two distinct modes of operations:

Passive: the cues are only sent as a suggestion to the nodes. Each node can decide whether or not to follow the guidelines suggested by the MCG. One particular example includes a suggestion that one node should decrease its general amplitude while another node should stay steady or a flashing warning indicating that the improvisation is about to switch to another section. Passive cues are generally rendered graphically as part of the score of the structured improvisation so that performers can take the suggestions (or obligations) into account while performing in remote sites.

Active: the cues are actively triggering/processing a concrete element on a distant node. This includes, amongst others, the opening of a filter or the interpolation of its center frequency, the reduction or augmentation of the amplitude of a distant node or the activation of a remote oscillator. Another aspect of active cueing that is currently being explored is not only the triggering of events from the MCG, but also the triggering of events from node to node. This possibility adds to the complexity of distributed cues and permits the building of complex patterns and interdependencies that use the network to create them.

A cue can be both passive and active simultaneously. One example would be the triggering of a sample along with the visual indication that a sample is about to be triggered. This introduces both an automated musical event (the sample) and an indication to a human performer that an event is about to occur, hence, suggesting a reaction of some sort.

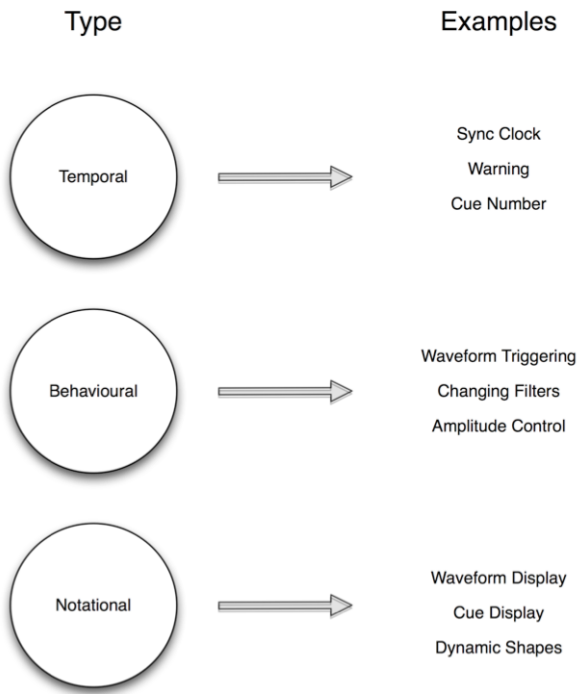


Figure 2. Cue types and corresponding examples

4. CHANGING TOPOLOGIES

The MCG should not be location centric by always being located in the same physical space, but should be able to take over a specific node at a given time. This leads to a far greater flexibility of the network topology as the MCG can virtually travel between nodes and position itself at any point on the network.

The option to change topologies means that, an NMP can start as a basic star network topology with one node being at the center of the network. In this case, the chosen node is not only at the center of the network but also takes the role of a leader in the performance. At any time the controlling node can transfer its powers to another node on the network. The move can happen when switching from a cue to another in the piece or it can be randomly attributed based on the distribution of roles and voting by other participants or the audience. Many permutations are possible, which lead ultimately to a change in the network topology. For example, in the case of a network with four nodes, called A, B, C and D respectively, if node C takes the lead, all the commands from the MCG will be issued by node C until the next change in topology. This series of permutations, as a network improvisation takes place, is analogous to the modus operandi of a musical improvisation that would happen on a real stage in terms of delegating control to a performer over others. This approach adds various levels of interplays between dislocated performers and leads to the creation of music that uses the network architecture as the core and to a certain extent as the score.

The concept of changing topologies outlined above allows the creation of complex interdependencies over the network (local or wide-area) and can easily implement some of the earlier network topology principles for musical collaboration such as the ones illustrated by Weinberg [11]. The MCG to node relationships allows the implementation of a, *"Process Centered Musical Network"* [12]. The flexibility brought by the MCG in terms of network topol-

ogy for the improvisation of musical content brings an additional layer of structure that is not necessarily musical, per se, but allows an allocation of performative roles despite the distance and the absence of physical contacts between the performers. This concept also ties up well with Weinberg's principles especially when the notion of Goal Oriented Interaction is mentioned [12].

The latter introduces two separate principles of interactions: Collaboration and Competition. The MCG along with structured improvisation allows the ensemble to morph between the notion of collaboration and the notion of competition, in musical terms, which creates diametrically opposed musical forms that are created by the changes in network topology.

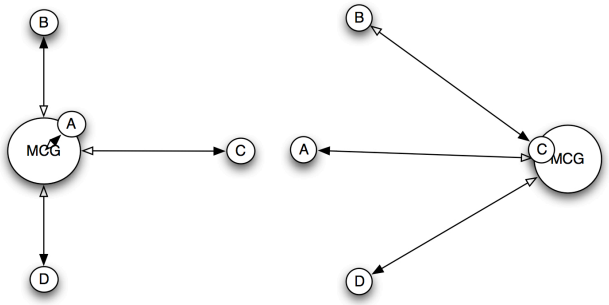


Figure 3. Changing relationship between nodes and MCG

5. THE ISSUE OF LATENCY

Latency is a pretty common term in the field of computer music and is defined as, *"the delay between the stimulus and the response"* [13]. In a more musical fashion and when parallelised with the speed of sound in air, latency can be defined as, *"the speed of sound through computer algorithms"* [10]. In the context of NMP, latency is often considered as a musical feature in its own right and, *"can be used as a specific compositional tool"* [13]. It needs to be highlighted that regardless of the quality and bandwidth of the networks used for NMPs the distance between two nodes will introduce a certain amount of latency. Even data traveling over fiber optic networks will be subject to a certain latency, not only because it cannot travel faster than the speed of light but also because that data will go through several switches and hubs along the way, introducing conversions and thus slowing down its delivery to destination. The MCG includes two approaches to latency, which are defined as synchronous interactions and asynchronous interactions.

5.1 Synchronous interactions

In this case the relationship between the MCG and the nodes is calculated in terms of time lag. For example, if the relationship from the MCG to node A is 100 milliseconds and the relationship from the MCG to node B is 75 milliseconds, an additional 25 milliseconds will be added to the relationship between the MCG and node B so that cues arrive at exactly the same time at node A and C. Since the networks used in this case are very stable, the timing is very likely to stay firm through the piece.

5.2 Asynchronous interactions

In this case, the MCG ignores the latency values between the MCG and the nodes and deals with the network as it is, leading to the generation of rhythmical patterns created by the network itself.

6. CONCLUSION AND FUTURE WORK

As illustrated through this paper, the MCG is the outcome of several years of practice in the field of NMP to answer the growing needs for distributed cueing structures. This ever-evolving exercise is an attempt to formalize some sort of convention in the practice of NMP and will be developed further as the practice progresses over time.

In the short to medium term, the goals regarding the development of the MCG and associated cueing strategies are:

- To develop a proper cross-platform application so that the system can be embraced by a wider community
- To offer a web platform on which a set of standard messages fitting in the cues classification outlined in this paper can be implemented and formalized by the NMP community
- To advertise in a more formal manner, mostly through online channels, applica-

tions and event in which the MCG is being used and can be further developed.

- To make the MCG freely available online to the NMP community.

7. REFERENCES

[1] Ascott, R.: *Telematic Embrace: Visionary Theories of Art, Technology, and Consciousness*. 1 edn. University of California Press, 2004.

[2] Bailey, D., 1993. *Improvisation: its nature and practice in music.* New York : Da Capo Press, 1993.

[3] Chafe, C., Wilson, S., Leistikow, R., Chisholm, D. and Savone, G., 2000. "A Simplified Approach to High Quality Music and Sound Over IP." *Proc. COSTG6 Conference on Digital Audio Effects (DAFx-00)*, Verona, 2000.

[4] Chafe, C., Gurevich, M., Leslie, G. and Tyan, S., 2004. "Effect of time delay on ensemble accuracy". *Proc. 2004 AES 117th Conf.,* San Francisco, 2004.

[5] Chafe, C., Wilson, S. and Walling, D., 2002. "Physical Model Synthesis with Application to Internet Acoustics." *Proc. 2002 Intl. Conference on Acoustics, Speech and Signal Processing,* Orlando, 2002

[6] Cycling '74. Available: http://www.cycling74.com [04/01/2008, 2008].

[7] Disparate Bodies. Available: http://www.sarc.qub.ac.uk/pages/db/ [04/20/2010, 2010].

[8] Net Vs. Net. Available: http://www.netvsnet.com [04/20/2010, 2010].

[9] OpenSoundControl. Available: http://opensoundcontrol.org/ [04/17/2010, 2010].

[10] Tanaka, A., 2000. "*Speed of Sound. In: J. BROUWER, ed, Machine Times"*. NAI Publishers, V2_Organisation.

[11] Weinberg, G., 2002. "The Aesthetics, History, and Future Challenges of Interconnected Music Networks", *Proc. 2002 International Computer Music Conference,* Göteborg, 2002.

[12] Weinberg, G., 2005. "Interconnected Musical Networks: Toward a Theoretical Framework." *Computer Music Journal* - Volume 29, Issue 2, pp. 23-29.

[13] Will, U.K., 2004. "Computer music modeling and retrieval", *International Symposium, CMMR 2003*, Montpellier, France, May 26-27, 2003: revised papers. Berlin ; Springer, c2004.

ISSUES AND TECHNIQUES FOR COLLABORATIVE MUSIC MAKING ON MULTI-TOUCH SURFACES

Robin Laney[1], **Chris Dobbyn**[1], **Anna Xambó**[1,2], **Mattia Schirosa**[2], **Dorothy Miell**[3], **Karen Littleton**[1], **Sheep Dalton**[1]
[1]The Open University, [2]Universitat Pompeu Fabra, [3]University of Edinburgh
{r.c.laney, c.h.dobbyn, a.xambo, k.s.littleton, n.dalton}@open.ac.uk, mattia.schirosa@upf.edu, d.e.miell@ed.ac.uk

ABSTRACT

A range of systems exist for collaborative music making on multi-touch surfaces. Some of them have been highly successful, but currently there is no systematic way of designing them, to maximise collaboration for a particular user group. We are particularly interested in systems that will engage novices and experts. We designed a simple application in an initial attempt to clearly analyse some of the issues. Our application allows groups of users to express themselves in collaborative music making using pre-composed materials. User studies were video recorded and analysed using two techniques derived from Grounded Theory and Content Analysis. A questionnaire was also conducted and evaluated. Findings suggest that the application affords engaging interaction. Enhancements for collaborative music making on multi-touch surfaces are discussed. Finally, future work on the prototype is proposed to maximise engagement.

1. INTRODUCTION

Applications for collaborative music making on multi-touch surfaces are an ideal setting for creative collaboration because they afford group participation and immediate music playing. According to Blaine and Fels [1], musical collaboration systems commonly restrict musical control, which facilitates novices' participation in the musical experience. The authors argue that the quality of the experience of using a collaborative music system takes precedence over the music produced: specifically that opportunities for social interaction, communication and connection with other partners is key to a satisfactory user experience. According to Mercer and Littleton [2], this provides us with a distinctive opportunity to foster learning and meaning-making through social interaction.

An analysis of the issues and techniques of music interfaces for multi-touch surfaces can provide us with a better understanding of these systems and help us to improve collaborative interaction. For that purpose, an exploratory user study was conducted and videotaped using a prototype application for a musical activity. Besides, qualitative evaluation was undertaken adapting Grounded Theory and Content Analysis. Quantitative evaluation was also undertaken based on questionnaire results. The main focus of the analysis is on the evaluation of the collaborative interactions enabled by the application. Enhancements for multi-touch applications for collaborative music making are discussed. Future work on the prototype is proposed to maximise engagement.

2. BACKGROUND

In this section we consider ways to address how people engage with technology in creative settings and musical multi-touch surfaces.

2.1 Creative engagement

There are numerous theoretical accounts of the nature of creative engagement with art and artefacts. Current models are based on a pragmatist view, which conceptualises the aesthetic and affective value of an object as lying not in the object itself, but in an individual's or a group's rich set of interactions with it [3, 4]. The phenomena of personal full immersion in an activity, also known as "flow" [4], has been extended to groups as means of heightening group productivity [5]. Facilitating productive conversation is a key way to achieve such "group flow". In the context of collective composition of music, Bryan-Kinns et al. [6] see attunement to others' contributions as the central principle of creative engagement.

2.2 Musical multi-touch surfaces

Musical tabletop applications are not new. A pioneering work is the ReacTable [7, 8], which allows a group of people to share control of a modular synthesizer by manipulating physical objects on a round table. The Music Table [9] uses a tangible interface based on cards representing notes or phrases laid on a table. Audiopad [10] uses the tracking of physical objects on a tabletop to access samples, cut between loops and carry out digital signal processing. Audiocubes [11] enables users to configure a signal processing network through the placement of physical cubes containing digital signal processors. In Xenakis [12], Markov Models are induced by placing stones on a tabletop interface. In contrast to all these systems, where movement of tangible objects is key, there are other systems centered on multi-touch interaction. Iwai's Composition on the Table [13] allows users to create music and visuals by interacting with four tables which display switches, dials, turntables and sliders. Stereotronic Multi-Synth Orchestra [14] uses

a multi-touch interface based on a concentric sequencer where notes can be placed. The work presented here is the design and evaluation of a simple multi-touch system which allows a group of people to create music by interacting using buttons. By keeping the system minimal, we are able to investigate the essential aspects of engaging interaction.

3. DESIGN CONSIDERATIONS

In this section we describe the properties and issues of multi-user instruments and of multi-touch systems.

3.1 Multi-user instruments

According to Jordà [15], multi-user instruments are tools that not only facilitate responsiveness and interaction between each performer and the instrument, but also between performers. The degree of interaction between performers is a key factor in achieving an engaging collective interplay. New digital instruments are especially suited for music collaboration because they are multi-process oriented supporting multiple and parallel musical processes [15] as well as their interface layouts being flexible enough to be able to exploit several strategies of collaboration by distributing the controls [16]. Thus, issues to be considered in multi-user instruments are shared versus local control as well as complexity versus simplicity.

3.1.1 Shared vs. local control

Shared controls, local controls or both are traditionally accommodated by collaborative multi-user instruments. Shared controls allow users to have a common display where the control is shared and can be the object of group discussion, whereas local controls consist in replicated controls which tend to be easier to reach for users and are identified with territoriality [16]. However, in the latter there is a design challenge when the number of replicated controls is large. The controls of a multi-user instrument may afford flexible or fixed number of performers; fixed or dynamic roles; and democratic or hierarchical relationships among users [15].

3.1.2 Complexity vs. simplicity

According to Blaine and Fels [17], collaborative musical interfaces engage social interaction. This facilitates both novices and experts to make music. A tradeoff should be considered between enabling virtuosity (appealing to advanced musicians who prefer to have free rein to exploit the expressivity of the instrument), and limiting the features offered to enable simplicity of use (important to enable novice musicians to participate easily).

3.2 Multi-touch systems

The properties of multi-touch systems are specially suited to the key design needs of musical instruments in general, and multi-user instruments in particular. In 1985, Buxton [18] introduced a set of properties, issues and techniques in multi-touch systems, later reviewed in 2007 [19], which have been applied to the question of multi-user musical instruments in this paper. In summary, the features considered are discrete versus continuous actions, size of display and context, sense of touch enabled and multiplicity of interaction opportunities.

3.2.1 Discrete vs. continuous

Multi-touch interaction can support both discrete and continuous actions. An example of a multi-touch interface using discrete actions would be an audio mixer, where one or more fingers push buttons or switches. An example of a continuous action could be also an audio mixer, but where one or more fingers move sliders, dials or knobs; or a waveform editor, where two fingers stretch a waveform.

3.2.2 Size and context

Display size is a decisive factor in how many fingers or hands can be used as well as how many performers can interact. Given that there is no mechanical intermediate device such as a mouse or a stylus, multi-touch systems are useful in tough environments such as classrooms or public spaces where these additional input devices can get damaged.

3.2.3 Sense of touch

Contact and position are traditionally those most used aspects of touch in multi-touch systems. Other features exploited are the degree of pressure; the angle of the finger relative to the surface; or the frictional force. However, there are some issues to be considered such as having lower precision than pointing with a stylus that can be solved, for example, by integrating physical controls with the interface [20, 16]. Besides, although the features of a physical input device are emulated, the interaction is actually with a virtual device where the visual supersedes other senses such as kinesthesia. As a result, users typically pay more attention to audiovisual feedback, which therefore should be reinforced in these systems.

3.2.4 Multiplicity

In multi-touch systems, some considerations to take into account are the following: first, even though the manipulation of a single point of contact can be exploited, multiple points are easier to use. Second, many of the interaction techniques from GUI (pointing at, dragging, clicking down or double clicking, for example) can be applicable in a multi-touch context using gestures based on discrete and continuous actions. Similarly, the same interaction technique can be split into multi-hand or multi-finger, depending on the granularity of the interaction sought. Lastly, where there are multiple users, the system should be able to distinguish the gestures and touches of the users from one another.

4. PROTOTYPE

In 2009, a project was undertaken in collaboration with the Milton Keynes Art Gallery under the theme of galleries and musical engagement. A prototype was developed with

the aim of enabling up to four users to collaborate on a composition.

The system needed to engage advanced musicians as well as novices given the emphasis on collaborative interaction. To this end, a simple prototype for a multi-touch tabletop was built where users could develop a musical composition using a palette of pre-composed samples. The system was populated with musical phrases in a traditional pop-contemporary musical style, emphasising harmony and rhythm, instead of complex melodic evolution (such as in jazz or classical music). Each user controlled a set of four graphical buttons which corresponded to four looping samples representing an instrument (bass line, drum line, keyboard line and percussion line). Each sample in the set consisted of a single musical phrase, consistent with all the others, which went from less to more complex. This evolution was shown in the buttons of the interface with a rounded shape from less to more filled. An additional button was also controlled by the user which toggled between either playing the sound through headphones in a private space or publicly through speakers (see Figure 1 the corresponding interface diagram). This switch icon changed shape and pulses when public mode was selected. The user was able to contribute by deciding which sample loop of his or her corresponding instrument was played and when it was triggered. Although each user could only play four samples, which meant there were only 256 combinations of loops, it represented an initial context for observing the processes of collaborative composition.

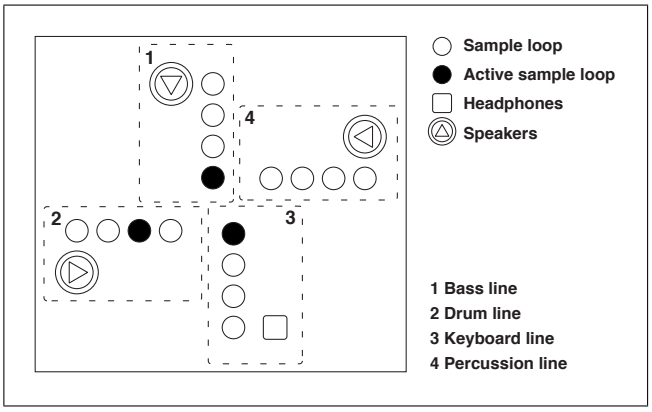


Figure 1. Interface diagram.

Regarding the hardware and software of the system, fingers are illuminated using diffuse infrared illumination and tracked with a camera underneath the table which encodes the information as a real time video stream. The reacTIVision [21] vision engine processes the video stream identifying the position of the finger tips. This data is encoded with the TUIO protocol over OSC and sent to a multi-touch software application (MTS). The MTS manages both visuals and audio: whereas the visual feedback, which is projected on to the surface, is defined using the programming language Processing [22]; the audio component is built using the visual programming language for music MAX/MSP [23].

5. METHOD

In this section a case study protocol is described which serves as a framework for an exploratory multi-case study. Afterwards, the qualitative approach to the analysis of the data is explained.

5.1 Exploratory multi-case study

A case study protocol was designed with the aim of studying the above prototype for collaborative music making. The approach was exploratory [24], in order to build an initial understanding of the situation. Video was chosen as the primary data source, given its advantages of multiple replay and closer multimodal analysis of interaction and also because the full range of behaviours and speech can be recorded easily, compared to other possible approaches [25]. Case studies of three groups were conducted in order to work with an initial phase of evaluation (see Figure 2).

Figure 2. Four users playing with the prototype.

The aim of the case study was to examine the extent to which the application enabled users to collaborate and the degree of mutual engagement in the creative process it afforded [6]. Involving participants with a range of levels of self-rated music knowledge was expected to provide a deeper insight into the situation because we would be able to examine the perspectives of both novices and experts. A number of features were observed: the ease of learning to use the application, the establishment of various roles for participants in the collaborative setting and how decision making was handled. Public spaces were ideal settings for carrying out this study for two main reasons: first, the physical proportions of these tabletop applications require ample space; and second, if they were to be used in performance and music learning, they would require shared spaces of similar size, with room for a group of participants and perhaps an audience. Thus, the user studies were conducted in a spacious atrium. Next, the musical tasks and questionnaire are described.

5.1.1 Musical tasks

With each group, three musical tasks of different character as well as an informal discussion were video-recorded in

order to generate sufficient data to analyse several aspects of behaviours using the prototype application. We were interested in any difficulties users might experience with the application, to what extent it enabled them to collaborate, and the degree to which it engaged them.

The tasks to be performed were the following:

1. T1. Initial period of sound exploration (3 min).
2. T2. Structured task with a score and a coordinator (7 min).
3. T3. Unstructured task of free improvisation (5-10 min).
4. T4. Informal discussion.

Each user had two signs with the messages of "sounds good" and "sounds bad" which could be raised at any moment of the performance.

Firstly, the explorative task was devoted to taking first steps with the tool. Participants were encouraged to switch between phrases, create solos and even make mistakes intentionally in order to learn how to control their own instrument. Secondly, the structured task was designed to produce a collaborative piece of music following a score (see Figure 3) and led by a coordinator who gave instructions during the interpretation of the musical score. The musical structure was built according to a traditional approach of musical dynamics, consisting of an introduction (two instruments), a crescendo (three instruments), a resolution (four instruments, but one is present sparingly), a diminuendo (three to two instruments), a more intensive crescendo (four instruments), a finale (four instruments) and, afterwards, a coda (decreasing one by one from three instruments). Each part lasted one minute, and a sign was given 30 seconds before the next move. In this task, the team was expected to decide which instruments should take part in each phase. This task was intended to help participants become accustomed to working together, in order to prepare them for the next task. Thirdly, an unstructured task of free improvisation without a coordinator and with no imposed rules was performed. In a more experimental fashion, the team was expected to decide not only the musical content but also the structure. Fourthly, an informal brief focus group discussion with the participants was held about the music compositions and how the music application could be improved.

5.1.2 Questionnaire

After that, a short text-based questionnaire was administered in order to collect information about the participants' profiles with questions about age, gender or previous music knowledge as well as the user experience with questions about the level of collaboration, difficulty, enjoyment or concentration.

5.2 Data analysis

By analysing the data from each set of observations, general patterns were extracted. These generalisations facilitate a better understanding of collaborative music making on touchable surfaces in particular, and collaborative interaction in general. For that purpose, two complementary qualitative approaches for data analysis were adapted: on the one hand, Grounded Theory, where open coding is applied to the data collected in a bottom-up fashion; and on the other hand, Content Analysis, where structured coding is identified taking a top-down perspective. By using two complementary analytical techniques a more rounded understanding could be achieved.

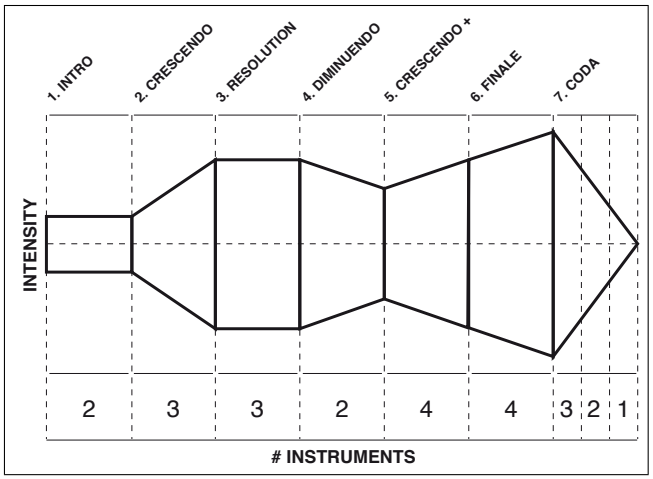


Figure 3. Graphical score of the structured task.

5.2.1 Open coding

Grounded Theory (GT) [24] is a qualitative research method employed in the social sciences that derives theoretical explanations from the data without having hypotheses in mind. In the initial stage of analysing the experiments, GT was adapted to offer a first insight to the data. According to this inductive procedure, the steps taken are: first, open coding of the video interactions identifying key moments (e.g. behaviours or opinions); second, grouping the codes by concepts; and third, generating general explanations from the categorisation of the concepts. Given that this approach is based on creative interpretation, we add more evidence by complementing GT with Content Analysis.

5.2.2 Structured coding

Content Analysis (CA) is defined by Holsti (1969) as "any technique for making inferences by objectively and systematically identifying specified characteristics of messages" [24]. This definition includes content analysis of text, videos, music or drawings. There are varied approaches to CA using quantitative, qualitative or both techniques. Our approach is derived from ethnographic content analysis or qualitative content analysis [26], an approach to documents that emphasizes the role of the investigator in the construction of the meaning of texts. The steps taken are the same as those explained in the open coding section, but with a difference in the first step: structured codes help us identify key points of the video-recorded interactions. The nomenclature is chosen from two existing theoretical frameworks. The first one is a general framework of tan-

gible social interaction [27]. The second one is focused on the engagement between participants in music collaboration [6].

6. FINDINGS

In this section we present the results obtained from the exploratory multi-case study. Participants, tasks, open coding, structured coding and the questionnaire are described.

6.1 Participants

We recruited 12 people. We conducted sessions with three groups, each with four participants, made up as follows:

- Group 1 (G1) contained fairly experienced musicians, with a combined level of skill (based on a self-assessment on a scale of 1 – 5) of 16;
- Group 2 (G2) comprised one experienced musician and three novices; the combined self-assessed rating was 8;
- Group 3 (G3) consisted of less musically adept participants, with a combined self-assessed rating of 9.

6.2 Tasks

All the sessions were videotaped. After each, we held an informal discussion with the participants around the table, talking through questions such as their feelings about the exercise, the quality of the composition they had evolved, and how the application could be improved.

In general, the three groups alternated between deciding some collaborative strategies before playing with deciding while playing. For example, the group with more advanced skills planned the unstructured task whereas the other two groups planned the structured task.

6.3 Findings from open coding

From transcription of the video speech and behaviours, and then the process of open coding, we identified the following concepts: collaboration, musical aesthetics, learning process and system design.

6.3.1 Collaboration

Collaboration in terms of awareness of other instruments was a challenge: "I think to be really aware of what we do we need to have maybe more time" (G1); "Should I concentrate on my own tempos or be aware of the other tempos?" (G2); "I think I hear all of them, maybe not the bass but it is fine" (G3). More visual feedback was requested: "I think it could be interesting to have a visual control of what is going on" (G1); "Adding a metronome, and maybe a different colour for the first bit, would help everyone to follow all the loops, the patterns" (G1); "We have to count each other and see what to do (...) a metronome at a right place" (G3).

In all three groups there is speech evidence of collaborative decision making before starting the musical tasks or while playing, with beginning sentences such as "I suggest, Shall we?, Should I?, Who's gonna?, Are we?, I think, Do we?, How about?, We can, Let's" (G1); "We can, I think, What do you think?, Let's, Can someone?, Can we?, You can" (G2) and "I suggest, Let's, Who?, Do you want? We can, Why don't you, I think, We could" (G3).

6.3.2 Musical Aesthetics

Emotiveness was expressed mostly with body gestures: all three groups voted regularly either "sounds good" or "sounds bad"; there were applauses at the end of the pieces (G1, G3) and one user even imitated a bass guitar player (G2).

Playfulness was conveyed with sentences such as "It was enjoyable" (G2); "I think I am having too much fun" (G3); and "It was very funny, I liked it a lot" (G2).

6.3.3 Learning process

The different parts of the structured task were understood with difficulty: "I found that it was difficult to figure out how to do the crescendo and the diminuendo" (G1); "Too much rules" (G2); and "I haven't understood what is the difference between finale and crescendo" (G3).

6.3.4 System design

The system responsiveness determined the expressivity: "The only difficulty that I had was switching" (G1); " I was too slowly, I'll try again" (G2) and "[When pressing a button] stays on and doesn't go off immediately was difficult" (G3).

Several improvements were suggested regarding individual expressivity with the presence of more features such as volume control (G1, G3); more samples (G1, G3); better responsiveness of the system (G3); a preview for the next sound to be played (G1) and a visual distinction between the active sound and the preview sound (G1). These features would provide more support to advanced users. Other suggestions were about improving the collaboration among the users with the presence of global shareable controls such as capability of modifying others (G3) or visual feedback such as the tempo (G1) or what other users where doing (G1). Another aspect commented on was how to improve the communication between users with the presence of virtual buttons for voting "sounds good" or "sounds bad" (G1) and also a big screen for visualising the music (G2).

Fun and social interaction were associated with the system. Possible contexts in public spaces were suggested such as a pub (G2) or a radio station (G2). Its use as a tool for composing was also mentioned (G3).

6.4 Findings from structured coding

Below we look at the concepts in [27] of tangible manipulation, spatial interaction, embodied facilitation or expressive representation (6.4.1 – 6.4.4) and the features in [6] of mutual awareness, shared and consistent representations, mutual modifiability and annotation (6.4.5 – 6.4.8). We found that some of the content analysed was already discussed in the open coding process (6.3), which provides consistency.

6.4.1 Tangible manipulation

The system provided a clear relationship between actions such as selecting a sample and effects such as listening to the sound selected: "It was not difficult (G1)"; "The technology was quite easy to get used to" (G3). However, rapid feedback during the interaction should be improved in terms of responsiveness (see quotes in 6.3.4), in order to facilitate expressivity and collaboration.

6.4.2 Spatial interaction

The space where the user studies were conducted facilitated a meaningful public space where people met and made music collaboratively with the system. However, the reciprocal fact of seeing and being seen could be improved with more visual cues of what was happening (see quotes in 6.3.4) The large size of the table with the display divided into four replicated controls allowed communication using body movement while interacting with the system (see quotes in 6.3.2).

6.4.3 Embodied facilitation

The set-up size as well as the form and location of the controls determined the way users collaborated. The options of manipulation were constrained to a single sound for each user. This aspect could be improved allowing multiple access points for each user: "Would be nice if you could play two [samples] at the same time" (G3). A representation built on users' experience should also be developed in order to connect not only with the skills of novices but also with experts (see quotes in 6.3.4).

6.4.4 Expressive representation

Users talked while interacting with the system, and they made decisions (see quotes in 6.3.1). Legibility of system reactions could be improved with visual feedback and better responsiveness (see quotes in 6.3.4).

6.4.5 Mutual awareness

The awareness of who was contributing and what they were contributing was difficult (see quotes in 6.3.1). This could be solved by strengthening with visual feedback the representation of both the identity of the contributor and of what kind of contribution it was. The awareness of where they were contributing was partial given that users only had individual controls and they reported difficulty in concentrating on both the individual contribution and the collaborative music piece (see quotes in 6.3.1).

6.4.6 Shared representations

A consistent view of the shared activity, independently of the user, is equivalent to the discussion conducted in spatial interaction of the previous theoretical framework (see quotes in 6.4.2).

6.4.7 Mutual modifiability

In a collaborative setting the presence of roles can imply an undesired hierarchical approach. The capacity of mutual modifiability, thus, can be a mechanism for having democratic roles. Thus, the system should incorporate this feature (see quotes in 6.3.4): "Sometimes I missed pushing the buttons of other people" (G3). That could avoid situations such as: "I feel that this one [keyboard] is having a lot of impact on the other sounds" (G1) or "[keyboard] is the most influencer" (G3).

6.4.8 Annotation

All tasks engaged conversation and the mechanism of voting specially contributed to supporting mutual engagement (see quotes in 6.3.2).

6.5 Findings from questionnaire

Data was also collected using a questionnaire, which was designed to probe such issues as how aware each participant had been of other instruments; the difficulty of the tasks, and how much they felt they had enjoyed and concentrated on them; and the extent to which they considered they had operated as a team and felt part of a collaborative process. Responses were recorded using numerical scores, but the questionnaire also asked for qualitative feedback on how participants organised themselves as a group and the nature of any rules they created. Although questionnaires were anonymous, we recorded the participants age, gender, previous experience, love of music, and the instrument they had been allocated on the table.

The questionnaire included the following five statements, with participants asked to give a score of between 1 and 5 (1 = strongly disagree; 5 = strongly agree).

- Q1. I felt we operated as a team.
- Q2. I felt part of a collaborative process.
- Q3. It was difficult to play.
- Q4. I enjoyed the music making task.
- Q5. I concentrated intensely on the music making task.

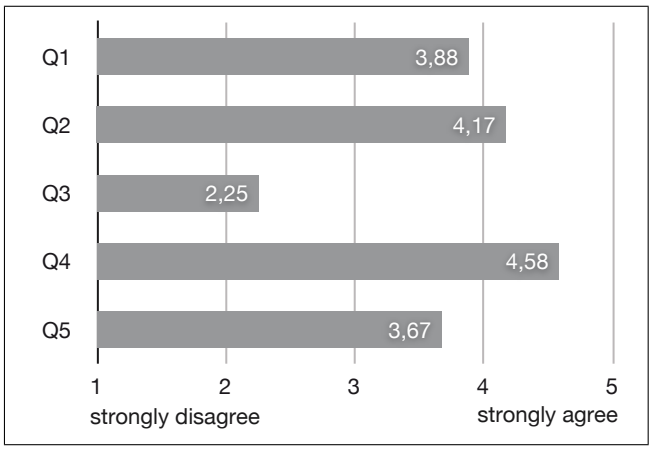


Figure 4. Averages for the 5 statements.

Satisfaction with the level of participation was generally high, with Q1 scoring an average of 3.88 and Q2 scoring

4.17. The difference between the team and collaborative scores may be of some interest, but was relatively small. No participant found the game especially difficult. To the Q3 statement the average response was 2.25. Participants reported high levels of enjoyment (Q4 average 4.58) and concentration (Q5 3.67). The fact that enjoyment and concentration were rated high and difficulty low is promising (see Figure 4).

7. DISCUSSION

Our appraisal of the three sessions focuses on three aspects of the groups performance:

1. the modes participants found to collaborate with one another;
2. difficulties that participants encountered and the extent to which they found the exercise engaging;
3. the degree of satisfaction at the end result. The perceived value of the musical product is obviously of importance.

In the actual sessions, four broad modes of interaction were used:

1. Visual. Participants were standing around the four sides of the table, and were thus able to look at each other.
2. Talking. Participants were free to address comments and suggestions to one another.
3. Auditory. Participants could choose to listen carefully to the patterns created by the other instruments and to concentrate on blending their own instrument with these.
4. Gestural. Participants could indicate suggestions and emotions by body motion.

As might be expected, groups used all four modes of interaction. Strategies of collaboration were suggested either before playing or during the music making tasks. Participants looked at one another consistently, in part probably influenced by the distribution of the setting. Similarly, the participants listened during all tasks, but with the support of other modes of interaction given the expressed difficulties in distinguishing each instrument. Body gestures were manifested constantly in pointing, voting "sounds good/bad", laughing or applauses.

Another interesting aspect of the groups possible collaboration was whether leaders emerged, or whether the collaborations were egalitarian. In Group 2, in particular, we anticipated that the experienced musician might take the lead. Perhaps surprisingly, in none of the groups did any dominant figure emerge, although one or another participant occasionally took the lead.

The findings of this study help us understand engagement in music collaboration. Qualitative video analysis and the questionnaires provide indication of participants having mutual engaging interaction in terms of being engaged with the music collaboratively produced and also being engaged with others in the activity. High degree of satisfaction at the end result is evidenced mostly by the gestural mode. The evidence found of participants exchanging ideas constantly indicates that the application strongly facilitates conversation, which, as noted earlier, is important in terms of group productivity. Within a user-centered design approach of active participation of users in the process of designing the system, the most two prominent aspects that have emerged as enhancements of multi-touch systems in music collaboration are:

- Responsiveness. The responsiveness determines the perceived emotiveness. This parameter should be adequately related to the system performance in terms of time and computer resources used. A consistent audiovisual feedback will enhance the perceived response of the system.
- Shared vs. individual controls. Both shared and individual spaces are needed. Shared features would strength mutual awareness and mutual modifiability. Individual spaces would strength personal opinion, musical identity and musical expression.

8. CONCLUSIONS AND FUTURE WORK

In this article, we described multi-user instruments and multi-touch systems, by enumerating their most prominent properties and issues. Then, we presented a simple and constrained prototype and explained the qualitative evaluation methodology undertaken in order to evaluate its creative engagement. Besides, we provided evidence of engagement and satisfaction with the end result. However, this initial exploratory case study should be complemented with a more formal study adding a control group in order to confirm that a minimal and constrained instrument as such can successfully engage. Finally, we proposed what design aspects should be considered in multi-touch surfaces for collaborative music making in order to support engagement. So far, this evaluation method not only provides us evidence of creative engagement but also an approach which can help us improve the prototype design.

We are interested in how many, and what type of, affordances such systems should offer in order to maximise engagement. At present the touchable nature of the table surface is not fully exploited, and there is scope to improve the responsiveness of the system and to redesign the distribution of shared versus individual controls. Furthermore, there is a plan to add individual continuous controls for sound parameter modifications in order to both encourage a process-oriented composition and improve engagement of advanced musicians. The mutual experience might be enhanced and collaboration deepened, by adding common controls – such as a metronome, through which global tempo could be displayed and altered. A balance between adding more features and keeping simplicity must be kept in order to attract both novices and experts alike.

9. ACKNOWLEDGEMENTS

The authors would like to thank all the colleagues of the Open University (Milton Keynes, UK) who have been involved in this study.

10. REFERENCES

[1] T. Blaine and S. Fels, "Collaborative musical experiences for novices," *Journal of New Music Research*, vol. 32, no. 4, pp. 411–428, 2003.

[2] N. Mercer and K. Littleton, *Dialogue and the Development of Children's Thinking*. Routledge, 2007.

[3] M. Blythe and M. Hassenzahl, *The semantics of fun: differentiating enjoyable experiences*, pp. 91–100. Kluwer Academic Publishers, 2004.

[4] M. Csikszentmihalyi, *Beyond Boredom and Anxiety: Experiencing Flow in Work and Play*. Jossey-Bass, 1975.

[5] K. Sawyer, *Group Genius: The Creative Power of Collaboration*. Basic Books, 2007.

[6] N. Bryan-Kinns and F. Hamilton, "Identifying mutual engagement," *Behaviour and Information Technology*, 2009.

[7] S. Jordà, M. Kaltenbrunner, G. Geiger, and R. Bencina, "The reacTable*," in *Proceedings of the 2005 International Computer Music Conference (ICMC '05)*, 2005.

[8] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, "The reacTable: Exploring the synergy between live music performance and tabletop tangible interfaces," in *Proceedings of the 1st international conference on Tangible and embedded interaction (TEI '07)*, pp. 139–146, ACM, 2007.

[9] R. Berry, M. Makino, N. Hikawa, and M. Suzuki, "The augmented composer project: The music table," in *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, p. 338, IEEE Computer Society, 2003.

[10] J. Patten, B. Recht, and H. Ishii, "Audiopad: a tag-based interface for musical performance," in *Proceedings of the 2002 conference on New interfaces for musical expression (NIME '02)*, pp. 1–6, 2002.

[11] B. Schiettecatte and J. Vanderdonckt, "Audiocubes: a distributed cube tangible interface based on interaction range for sound design," in *Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08)*, pp. 3–10, ACM, 2008.

[12] M. Bischof, B. Conradi, P. Lachenmaier, K. Linde, M. Meier, P. Pötzl, and E. André, "Xenakis: combining tangible interaction with probability-based musical composition," in *Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08)*, pp. 121–124, ACM, 2008.

[13] T. Iwai, "Composition on the table," in *International Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques, ACM, 1999.

[14] http://www.fashionbuddha.com/, 15/3/2010.

[15] S. Jordà, "Multi-user instruments: models, examples and promises," in *Proceedings of the 2005 conference on New interfaces for musical expression (NIME '05)*, pp. 23–26, 2005.

[16] R. Fiebrink, D. Morris, and M. R. Morris, "Dynamic mapping of physical controls for tabletop groupware," in *Proceedings of the 27th international conference on Human factors in computing systems (CHI '09)*, pp. 471–480, ACM, 2009.

[17] T. Blaine and S. Fels, "Contexts of collaborative musical experiences," in *Proceedings of the 2003 conference on New interfaces for musical expression (NIME '03)*, pp. 129–134, 2003.

[18] W. Buxton, R. Hill, and P. Rowley, "Issues and techniques in touch-sensitive tablet input," in *Proceedings of SIGGRAPH '85*, pp. 215–224, ACM, 1985.

[19] B. Buxton, *Multi-Touch Systems that I Have Known and Loved*. Microsoft Research, 2007.

[20] Y. Rogers, Y.-K. Lim, and W. R. Hazlewood, "Extending tabletops to support flexible collaborative interactions," in *Proceedings of the 1st International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP '06)*, pp. 71–78, IEEE Computer Society, 2006.

[21] M. Kaltenbrunner and R. Bencina, "reacTIVision: A computer-vision framework for table-based tangible interaction," in *Proceedings of the 1st international conference on Tangible and embedded interaction (TEI '07)*, pp. 69–74, ACM, 2007.

[22] C. Reas and B. Fry, *Processing: A Programming Handbook for Visual Designers and Artists*. MIT Press, 2007.

[23] M. Puckette, "Max at seventeen," *Computer Music Journal*, vol. 26, no. 4, pp. 31–43, 2002.

[24] J. Lazar, J. Feng, and H. Hochheiser, *Research Methods in Human-Computer Interaction*. Wiley, 2010.

[25] B. Jordan and A. Henderson, "Interaction analysis: Foundations and practice," *Journal of the Learning Sciences*, vol. 4, no. 1, pp. 39–103, 1995.

[26] D. L. Altheide, "Ethnographic content analysis," *Qualitative Sociology*, vol. 10, pp. 65–77, 1987.

[27] E. Hornecker and J. Buur, "Getting a grip on tangible interaction: A framework on physical space and social interaction," in *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06)*, pp. 437–446, ACM Press, 2006.

TOWARDS A PRACTICAL APPROACH TO MUSIC THEORY ON THE REACTABLE

Andrea Franceschini
Università di Padova
andrea.franceschini@gmail.com

ABSTRACT

This paper builds upon the existing Reactable musical platform and aims at extending and improving its approach to music theory [1] . Sections 1 and 2.2 explain the motivations that led to the development of this proposal from a musical point of view while also giving a music education perspective. In section 2 we'll see a brief survey on tabletop and tangible multi-user systems for audiovisual performance and we'll also briefly introduce the process of implicit learning, we'll formulate a hypothesis about music as a natural language, and describe how the work hereafter presented can help music education. In section 3 we'll describe the current state of the art about music theory on the Reactable, followed by an original proposal about a way to extend and improve it. Finally we'll see how people who had a chance to test the system found it interesting and playful, while also giving important feedback that can be used to improve many practical aspects of the implementation.

1. INTRODUCTION

The Reactable is a digital musical instrument with a multi user tabletop and tangible interface, designed to explore and perform experimental electronic music giving users the highest possible degree of freedom. Therefore it is a precise design choice to give it no knowledge of any form of music theory.

As the Reactable became widely known, it attracted interest from both experimental and traditional musicians, these latter complaining about the lack of a way to include "traditional" music in a performance, where "traditional" music means melodies made of notes.

In response to this, a set of objects was developed. This set includes a "sequencer" (fig. 2) that pilots waveform generators by telling them which notes of the western chromatic scale to play, and an object called "tonalizer" (fig. 1) that constrained all waveform generators to play only a limited set of pitches, from the whole chromatic scale to its subsets.

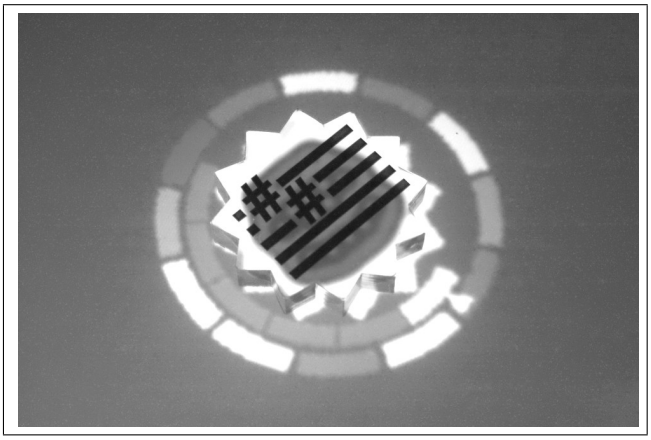


Figure 1. The current tonalizer.

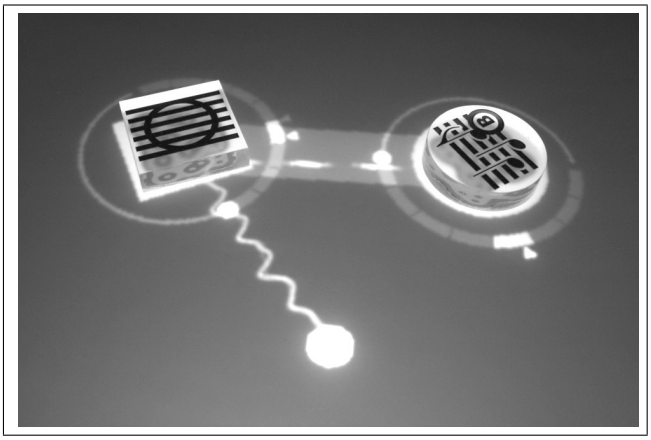


Figure 2. The current sequencer "playing" a plucked string instrument synthesizer.

Since traditional music was not a priority for the Reactable, these two objects were developed to the minimum level of functionality. For example, melodies have to be stored in advance and can only be selected by rotating the object. On the other hand, the tonalizer allows for "live" setting of a number of presets, but it basically allows to select any note on the scale, without any form of correction or automatic suggestion, thus requiring performers to have a certain level of music knowledge. Given tonalizer's target audience, such an assumption seems reasonable, but it can have unexpected – possibly unpleasant – results when used by unexperienced performers.

[1] **Disclaimer:** the results here presented reflect my personal research and opinions on the topic and, although being done in collaboration with the creators of the Reactable, they don't necessarily express the opinions and visions of the original team. Therefore all the developments I proposed are to be considered as a possible direction to be thoroughly examined, evaluated and validated through experimentation involving potential users of the system.

In section 2.1 we'll briefly cover the history of audiovisual and tangible interfaces, starting from the first experiments in audiovisual performance, to the most recent developments in computer-based tangible and multi-touch interfaces for sound and music manipulation. Section 2.2 presents a perspective about implicit learning, arguing how music can be considered a natural language, thus the knowledge about language learning can be applied to music in a similar way. This will be the key point in claiming that the work presented in section 3 is an approach to music theory on the Reactable, and eventually an effective aid to explicitly learn abstract music theory concepts, as we shall see in section 4.2. Finally, section 4.1 will present an overview of the main results that emerged from preliminary tests with users, covering both strong and weak points that will guide the future development phases.

2. BACKGROUND

2.1 Audiovisual and Tangible interfaces

As a tabletop and tangible interface for music performance with visual feedback, the Reactable puts together ideas that date quite back in time. If we think of a visual analogous to music, probably the earliest known machinery was built in 1734 by Louis-Bertrand Castel [1]. Later examples, which appeared during the early twentieth century, were the Clavilux by Thomas Wilfred (1919), and the Lumigraph by Oskar Fischinger (1955).

In recent years computers became more and more involved in music performance and production, and a number of different programs and interfaces were developed. One interesting example is Music Mouse by Laurie Spiegel (1985), which is a software intended to turn a personal computer into a musical instrument capable of being performed live by one user. In fact it turns motion of the computer's mouse into harmony and melody patterns, thus requiring virtually no music knowledge to the user, whom in turn can entirely focus on direction of the performance. Another interesting example is Instant Music by Electronic Arts (1986), which is a software explicitly aimed at musicians to assist them in creating original music, or even support them in a live performance on other instruments. The software allows users to "draw" melodies with no apparent limit, while a background correction process ensures that no "wrong notes" are played. Once again we have a system that applies harmony rules to allow users with even limited experience to proficiently create music.

Last but not least, tabletop and tangible computing has received lots of attention in the last decade, but the concept itself can be tracked earlier in time, for example in popular science fiction. An early notable example of Tangible User Interface is the "Urban Planning Workbench" [2], but more music-oriented works exist, such as the Jam-O-Drum [3], a gaming platform for up to twelve contemporary players, and the Audiopad [4], a musical instrument that replicates a modular synthesizer using RFID-tagged pucks.

As it is common in modern translucent tabletop interfaces, the Reactable also employs multi-touch interaction, thus allowing to perform gestures with fingers, other than with tangible objects. This makes it possible to develop interactive visuals that arguably allow to control many parameters using little space and a "familiar" interface [2].

2.2 Music education

Implicit learning is the process through which an individual becomes sensitive to the underlaying regularities of highly structured systems, like language or music. Even if such knowledge remains at a level such that one is not able to explicitly describe the rules, it influences perception and interaction with the environment [5]. The most prominent real-life example is natural language. Babies learn to speak at an early age by imitation, then later they explicitly learn why and how concatenation of some particular sounds conveys a meaningful message. More information about this topic is provided in [5] and [6].

It's been argued that the origin of music itself may be similar to that of natural languages [7]. If we think of music as a natural language then we can suppose it comes with its own set of symbols, words and sentences, all tied together by a grammar, a set of rules of a given harmony system. In this sense, each harmony system is a different natural language as much as English and Italian are. Also, generative grammar approaches have been used in musicology to analyze musical pieces [8] though the idea of a "universal grammar" [3] has not received much consensus while evidence of author specific, or even period specific, grammars is much more accepted.

In the hypothesis of music as a natural language, we can go further into assuming the existence of an associated transformational grammar through which it is not only possible to understand new and meaningful sentences, but also to produce them. More information about such grammars and their relationship with languages is provided in [9].

Assuming this hypothesis holds for music, we may start to see how the Reactable, with the modifications hereafter proposed, can prove [4] to be a valuable aid to music education. In fact, music is usually taught through teaching a musical instrument. While this is actually something a music student expects, it is also a time and resource intensive process. Moreover, empyrical evidence among music instructors seems to suggest that learning a second, possibly quite different, instrument when a first one is already mastered, is usually easier. The most likely explanation seems to be that a student approaching a second instrument already knows abstract concepts of music theory, so he or she may find it easier to relate to a new instrument when already knowing what it is to be musically expected. Despite

[2] Of course most of the parameters can be controlled with other tangible objects – in fact some of them are. Nonetheless some metaphores are more appropriate in some cases, for example users tend to be more used to a button other than the presence or absence of some specific tangible object when it comes to determine an on/off state of some sort.

[3] The term "universal" most reasonably refers to all music compositions under a single harmony system instead of a grammar that describes all the possible harmony systems, the latter being a fascinating hypothesis, though still unproven.

[4] It shall be clear that this is a proposal that can be taken into consideration only after the system is fully developed and evaluated, and its usability is strongly assessed. In fact such assessment requires an extensive experimentation phase involving music teachers and students on many levels, therefore the instrument is required to not pose significant difficulties that can invalidate the eventual findings.

seeming reasonable, this hypothesis merely comes from empyrical observations and so far it seems it's not been formally assessed [5] .

2.2.1 The Reactable as a learning aid

As it's detailed in [6], regularities and relations between tones, scales, chords, etc, are important when it comes to implicit learning, and learning in general. This is the key point being used argue that learning western tonal music can be optimised and improved using multimedia tools that emphasize such structures.

As we shall see in the next section, this paper's proposal can be effectively turned into such a system by integrating notions of musical structures and presenting them as an optional operating mode. In fact the system we're going to describe, together with the whole Reactable platform, is intended to be a non-intimidating, easy and playful musical instrument that can give students the ability to experience musical concepts by reducing the complexity of learning a traditional musical instrument, while leveraging on the implicit experience of music theory one may have unconsciously acquired.

It's finally worth noting that even if [6] only analyses western music theory, it can be argued that regularities and relations between other cultures' notions of musical structures exist, though they can be quite different from those existing in western music. However, if a tool is properly designed to be flexible and extendable enough, it should also be easy to adapt it to different rules, and this is one of the main goals that drove the design process we'll briefly see in the next section.

3. DESIGN AND IMPLEMENTATION

In section 1 we've briefly seen the current implementation of tonalizer and sequencer objects. Without further ado, let's examine the new implementation proposed in this work. While this paper only focuses on the finished proposal, an in-depth discussion of the proposal and design process that led to it can be found in [10].

3.1 Tonalizer

The "new" tonalizer was initially quite different from the original object, both visually and conceptually. It started as a round object surrounded by an arbitrary number of concentric rings, each divided into 12 sectors – one per tone in the chromatic scale, each tone representing a chord by its tonic. Each ring could be rotated in order to align different tones, thus allowing users to activate them by drawing a single stroke passing over each of them. This design had the potential to express progressions and transpositions, and it also could have worked as a melodic sequencer, except for the fact that it could potentially take a very large space on the Reactable. Therefore the original idea of chord slots came back and, with appropriate modifications, the design in figure 3 was eventually chosen.

[5] Such assessment is obviously far beyond the scope of this paper, but it can be taken into further consideration as a future development.

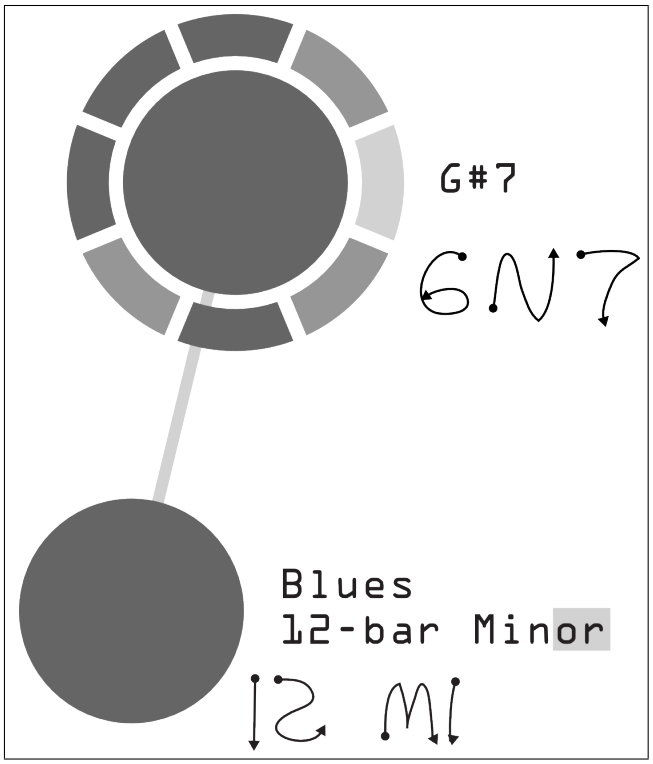


Figure 3. The new tonalizer.

The design depicted in figure 3 features new tonalizer's most complex configuration, the simplest being only made by the top round object and its surrounding button ring. While the original tonalizer uses a similar ring to choose which notes are to be allowed, this proposal uses it as a storage for "chord presets". A chord preset is an usual chord, such as the G#7 shown in the figure. This chord is used to derive a number of scales whose notes can be played along with the chord without resulting in "unpleasant" combinations. Users may then select which scale should be used with that particular chord, and from that moment on the notes of that scale become the only playable notes by any other waveform generator. Furthermore, some sort of "progression" object – the bottom one in figure 3 – can be used to produce an entire progression of chords starting with the chosen one and following the chosen specification – in this example a 12-bar minor blues progression in G# should be selected, and eventually "played" according to an external timing source.

Last but not least, this proposal introduces the concept of handwriting recognition. This feature was introduced as a compact yet powerful way to create chord presets. As we will see in section 4, it turned out that users liked the idea, and mostly found it funny and helpful – even if the actual implementation had some glitches and didn't always work as expected.

3.2 Sequencer

After the "unified" design was discarded – as described in 3.1 – the idea of a piano-roll like interface was immediately considered. In fact this idea went under a number of feature additions remaining almost untouched in its visual

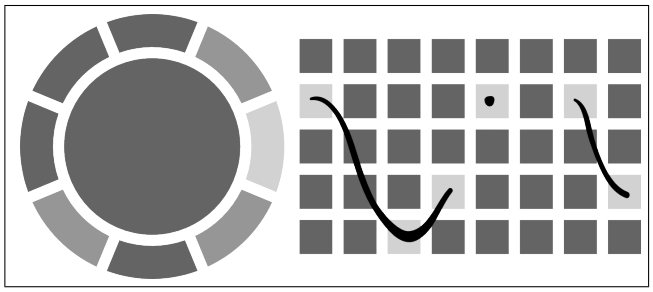


Figure 4. The new sequencer in editing mode with some gestures being performed.

appearance.

Figure 4 depicts the proposed melodic sequencer while editing a sequence. Like the tonalizer, it has a button ring around it as a sequence preset storage. On the other hand, unlike the current sequencer, these presets can be modified on the fly during a performance, rather than being preloaded. This is performed using the grid that's shown in the figure. The x axis represents time, while the y axis represents the notes of the scale that's currently chosen – for example, the scale in the figure may be a major pentatonic so, assuming the chord is the one of figure 3, the notes would be G#, A#, B#, D# and E#. Strokes and taps are used to turn on and off notes in the sequence. While a simple tap results in the obvious trigger of a note, the way a stroke acts is a bit more complex. In figure 3 we see that notes are triggered on some special points, namely stroke's extremes and "zero derivative" points – where the "derivative" is considered relatively to the x axis of the grid. Arguably some other cues can be used, such as speed of the finger, but this initial set proved to be sufficient most of the times.

4. RESULTS AND FUTURE DEVELOPMENT

4.1 Testing and evaluation

The whole design process went through a series of iterations which subsequently integrated suggestions from people familiar with the Reactable itself and with usability and HCI topics. In addition to that, two informal sessions with users were performed. The first one involved a few people who were familiar with the Reactable and HCI topics, and it was performed using the actual Reactable hardware running a proof-of-concept implementation of this proposal. The second one has been performed with users who were mostly unaware of both the Reactable and HCI, but with a basic to high level of music knowledge. This second phase was conducted with a slightly modified version of the proof-of-concept implementation running in a simulator, since the actual hardware was not available. Both groups of people were told about the Reactable, the purpose of the system here presented and its basic concepts – such as handwriting recognition and stroke-based composition – but not how to actually perform tasks such as chord and melody creation. They were told to perform a series of tasks, from creating a chord preset to composing a simple melody, and their reactions were recorded.

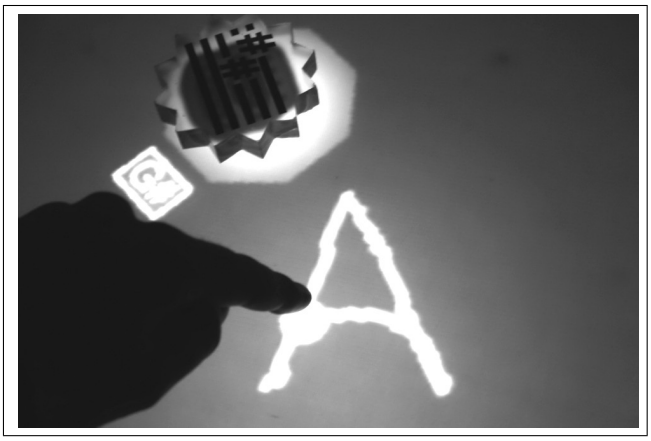


Figure 5. The action of drawing the letter A. Since there's no edit action associated with letter A, a new A major chord is going to be created.

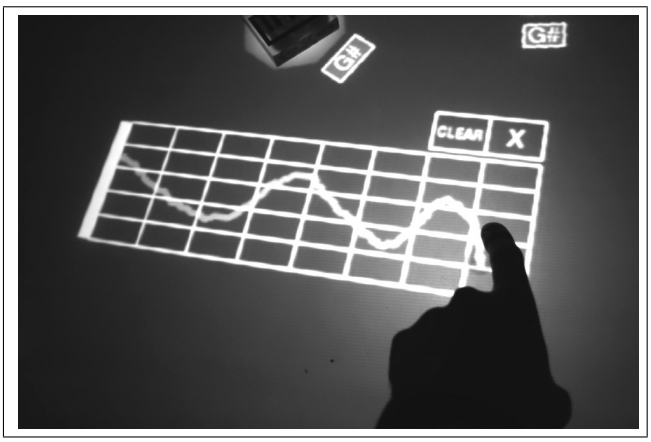


Figure 6. The piano-roll like interface with the stroke gesture used prior to analysis.

Although few data was gathered, these two phases reported mixed yet interesting results.

- Most of the people familiar with the Reactable regarded this proposal as an interesting development, mostly because of the possibility to choose a subset of notes that ensures that no mistakes are made, while also doubting that this whole renewed Tonalizer could really add some significant value to the Reactable as an instrument. On the other hand, some of those who weren't familiar enough with the Reactable didn't always get how this was an improvement at all, being just more fascinated with the original Reactable and its sound exploration freedom.

- Regarding the overall simplicity of task performance, most of the people – both familiar and not – reported that some actions weren't that obvious to perform, for example the gesture that opens the piano-roll (figure 6).

- They also reported that the reason because some slots around the tangibles were filled or empty was not really clear, although finding it reasonable when told.

This suggests that a more expressive visual feedback may be developed for greater clarity.

- Almost everyone noticed that the piano-roll didn't always work as expected. This was absolutely expected due to the unrefined implementation of the algorithms.

- Nonetheless almost everybody found the handwriting idea (figure 5) pretty interesting in perspective, even funny, although it didn't always work as expected, but this is again due to unrefined implementation.

All these observations suggest a number of practical improvements that'll be addressed in the future. For a complete review of the improvable aspects of this proposal, see [10].

4.2 Future developments about different musical cultures and music education

Western tonal music, not unlike many other musical systems, features relations between chords, scales and tones. The key point that would make this proposal a valuable aid to music education is the integration of such structures into the two proposed objects. Thinking in the western tonal music framework, a Tonalizer that can communicate concepts like the circle of fifths or the relations between chords in a progression, and a Sequencer that highlights whose tones are stable, whose are passing, and whose are consonant/dissonant, would be helpful in internalizing music theory.

However, as already hinted in subsection 2.2, not all the musical systems rely on the same concepts found in western music theory. The integration and effective conveyance of these different sets of rules using the objects proposed in this paper is a challenging development that would involve an extensive study of the musical systems to be integrated, followed by the design of a proper interface that can proficiently help to understand the desired concepts.

There is a final note that's worth making about subsection 2.2. Music education is not the entire story. In fact, during the development process there had been contacts with people involved in education and assistance to people with disabilities, such as physical handicaps, or even autism and learning disorders. None of them knew about the Reactable, yet it extremely fascinated most of them as a tool to make disabled people approach music and possibly help them express themselves. Though extremely interesting, this is far beyond the scope of this work. Nonetheless, with a more developed and assessed system, further investigation may be possible.

5. CONCLUSIONS

In this paper we've seen how the Reactable's existing approach to pitch-based music can be extended and eventually generalized to approach music theory in a broader sense. Even if not all the possible directions and ideas have been implemented and tested, they will be addressed in the future.

The objects here presented proved to work [6] and people involved in informal testing sessions expressed interest in the project and provided useful feedback to start a new development phase.

An extensive formal testing and assessment phase will also be required, first during the development process, and second after the system is mature enough to start experimenting in real world music education situations.

6. REFERENCES

[1] G. Levin, "Painterly interfaces for audiovisual performance," Master's thesis, Massachussets Institute of Technology, 2000.

[2] J. Underkoffler and H. Ishii, "Urp: A luminous-tangible workbench for urban planning and design," in *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pp. 386–393, ACM, 1999.

[3] T. Blaine and T. Perkis, "The Jam-O-Drum interactive music system: A study in interaction design," in *Proceedings of the ACM DIS 2000 Conference*, ACM Press, August 2000.

[4] J. Patten, B. Recht, and H. Ishii, "Audiopad: a tag-based interface for musical performance," in *Proceedings of the 2002 conference on New Interfaces for Musical Expression*, May 2002.

[5] C. A. Seger, "Implicit learning," in *Psychological Bullettin*, pp. 115:163–169, 1994.

[6] E. Bigand, P. Lalitte, and B. Tillmann, *Sound to Sense, Sense to Sound. A State of the Art in Sound and Music Computing*, ch. 2. Logos, 2008.

[7] J. Molino, "Toward an evolutionary theory of music and language," in *The Origins of Music* (N. L. Wallin, B. Merker, and S. Brown, eds.), Bradford books, pp. 165–176, The MIT Press, 2000.

[8] N. Ruwert and M. Everist, "Methods of analysis in musicology," in *Music Analysis*, vol. 6, pp. 3–36, Blackwell Publishing, 1987.

[9] N. Chomsky, *Aspects of the Theory of Syntax*. The MIT Press, 1965.

[10] A. Franceschini, "A practical approach to Music Theory on the Reactable," Master's thesis, Università di Padova, 2010.

[11] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, "The reacTable: Exploring the synergy between live music performance and tabletop tangible interfaces," in *Proceedings of the first international conference on "Tangible and Embedded Interaction" (TEI07)*, Baton Rouge, Louisiana, 2007.

[6] An early demonstration performed on the actual Reactable hardware is available at `http://vimeo.com/4325822`.

[12] M. Bosi, “Extending physical computing on the Reactable,” Master’s thesis, Universitat Pompeu Fabra, 2009.

TOWARDS ADAPTIVE MUSIC GENERATION BY REINFORCEMENT LEARNING OF MUSICAL TENSION

Sylvain Le Groux
SPECS
Universitat Pompeu Fabra
sylvain.legroux@upf.edu

Paul F.M.J. Verschure
SPECS and ICREA
Universitat Pompeu Fabra
paul.verschure@upf.edu

ABSTRACT

Although music is often defined as the "language of emotion", the exact nature of the relationship between musical parameters and the emotional response of the listener remains an open question. Whereas traditional psychological research usually focuses on an analytical approach, involving the rating of static sounds or preexisting musical pieces, we propose a synthetic approach based on a novel adaptive interactive music system controlled by an autonomous reinforcement learning agent. Preliminary results suggest an autonomous mapping from musical parameters (such as tempo, articulation and dynamics) to the perception of tension is possible. This paves the way for interesting applications in music therapy, interactive gaming, and physiologically-based musical instruments.

1. INTRODUCTION

Music is generally admitted to be a powerful carrier of emotion or mood regulator, and various studies have addressed the effect of specific musical parameters on emotional states [1, 2, 3, 4, 5, 6]. Although many different self-report, physiological and observational means have been used, in most of the cases those studies are based on the same paradigm: one measures emotional responses while the subject is presented to a static sound sample with specific acoustic characteristics or an excerpt of music representative of a certain type of emotions.

In this paper, we take a synthetic and dynamic approach to the exploration of mappings between perceived musical tension [7, 8] and a set of musical parameters by using Reinforcement Learning (RL) [9].

Reinforcement learning (as well as agent-based technology) has already been used in various musical systems and most notably for improving real time automatic improvisation [10, 11, 12, 13]. Musical systems that have used reinforcement learning can roughly be divided into three main categories based on the choice of the reward characterizing the quality of musical actions. In one scenario the reward is defined to match internal goals (a set of rules for instance), in another scenario it can be given by the audience (a like/dislike criterion), or else it is based on some notion of musical style imitation [13]. Unlike most previous examples where the reward relates to some predefined musical rules or quality of improvisation, we are interested in the emotional feedback from the listener in terms of perceived musical tension (Figure 1).

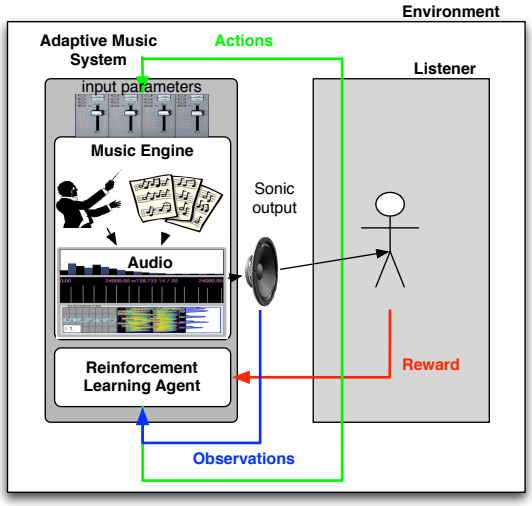


Figure 1. The system is composed of three main components: the music engine (SiMS), the reinforcement learning agent and the listener who provides the reward signal

Reinforcement learning is a biologically plausible machine learning technique particularly suited for an explorative and adaptive approach to emotional mapping as it tries to find a sequence of parameter change that optimizes a reward function (in our case musical tension). This approach contrasts with expert systems such as the KTH rule system [14, 15] that can modulate the expressivity of music by applying a set of predefined rules inferred from previous extensive music and performance analysis. Here, we propose a paradigm where the system learns to autonomously tune its own parameters in function of the desired reward function (musical tension) without using any a-priori musical rule.

Interestingly enough, the biological validity of RL is supported by numerous studies in psychology and neuroscience that found various examples of reinforcement learning in animal behavior (e.g. foraging behavior of bees [16], the dopamine system in primate brains [17], ...).

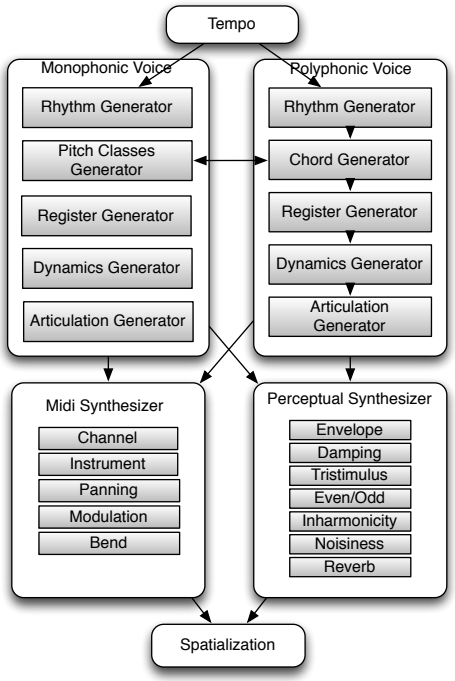


Figure 2. SiMS is a situated music generation framework based on a hierarchy of musical agents communicating via the OSC protocol.

2. A HIERARCHY OF MUSICAL AGENTS FOR MUSIC GENERATION

We generate the music with SiMS/iMuSe, a Situated Intelligent Interactive Music Server programmed in Max/MSP [18] and C++. SiMS's affective music engine is composed of a hierarchy of perceptually meaningful musical agents (Figure 2) interacting and communicating via the OSC protocol [19]. SiMS is entirely based on a networked architecture. It implements various algorithmic composition tools (e.g: generation of tonal, Brownian and serial series of pitches and rhythms) and a set of synthesis techniques validated by psychoacoustical tests [20, 3]. Inspired by previous works on musical performance modeling [14], iMuSe allows to modulate the expressiveness of music generation by varying parameters such as phrasing, articulation and performance noise.

Our interactive music system follows a biomimetic architecture that is multi-level and loosly distinguishes sensing (the reward function) from processing (adaptive mappings by the RL algorithm) and actions (changes of musical parameters). It has to be emphasized though that we do not believe that these stages are discrete modules. Rather, they will share bi-directional interactions both internal to the architecture as through the environment itself [21]. In this respect it is a further advance from the traditional separation of sensing, processing and response paradigm[22] which was at the core of traditional AI models.

In this project, we study the modulation of music by three parameters contributing to the perception of musical tension, namely articulation, tempo and dynamics.

While conceptually fairly simple, the music material generator has been designed to keep the balance between predictability and surprise. The real-time algorithmic composition process is inspired by works from minimalist composers such as Terry Riley *(In C, 1964)* where a set of basic precomposed musical cells are chosen and modulated at the time of performance creating an ever-changing piece.

The choice of base musical material relies on the extended serialism paradigm. We a priori defined sets for every parameter (rhythm, pitch, register, dynamics, articulation). The generation of music from these sets is then using non-deterministic selection principles, as proposed by Gottfried Michael Koenig [23]. (The sequencer modules in SiMS can, for instance, choose a random element from a set, or choose all the elements in order successively, choose all the elements in reverse order, or play all the elements once without repetition, etc.)

For this project we used a simple modal pitch serie [0, 3, 5, 7, 10] shared by three different voices (2 monophonic and 1 polyphonic). The first monophonic voice is the lead, the second is the bass line, and the third polyphonic voice is the chord accompaniment. The rhythmic values are coded as 16n for a sixteenth note, 8n for a eighth note, etc. The dynamic values are coded as midi velocity from 0 to 127. The other parameters correspond to standard pitch class set and register notation. The pitch content for all the voices is based on the same mode.

- Voice1:
  - Rhythm: [16n 16n 16n 16n 8n 8n 4n 4n]
  - Pitch: [0, 3, 5, 7, 10]
  - Register: [5 5 5 6 6 6 7 7 7]
  - Dynamics: [90 90 120 50 80]
- Voice2:
  - Rhythm:[4n 4n 4n 8n 8n]
  - Pitch: [0, 3, 5, 7, 10]
  - Register: [3 3 3 3 4 4 4 4]
  - Dynamics: [90 90 120 50 80]
- Polyphonic Voice:
  - Rhythm: [2n 4n 2n 4n]
  - Pitch: [0 3 5 7 10]
  - Register: [5]
  - Dynamics: [60 80 90 30]
  - with chord variations on the degrees [1 4 5]

The selection principle was set to "series" for all the parameters so the piece would not repeat in an obvious way [1]. This composition paradigm allows the generation of constantly varying, yet coherent, musical sequences. Properties of the music generation such as articulation, dynamics modulation and tempo are then modulated by the RL algorithm in function of the reward defined as the musical tension perceived by the listener.

[1] Samples: http://www.dtic.upf.edu/~slegroux/confs/SMC10

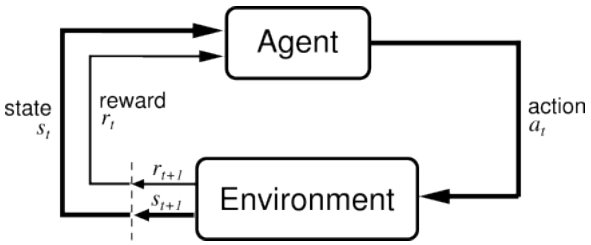


Figure 3. The agent-environment interaction (from [9])

3. MUSICAL PARAMETER MODULATION BY REINFORCEMENT LEARNING

3.1 Introduction

Our goal is to teach our musical agent to choose a sequence of musical gestures (choice of musical parameters) that will increase the musical tension perceived by the listener. This can be modeled as an active reinforcement learning (RL) problem where the learning agent must decide what musical action to take depending on the emotional feedback (musical tension) given by the listener in real-time (Figure 1). The agent is implemented as a Max/MSP external in C++, based on RLKit and the Flext framework [2].

The interaction between the agent and its environment can be formalized as a Markov Decision Process (MDP) where [9]:

- at each discrete time t, the agent observes the environment's state $s_t \in S$, where S is the set of possible states (in our case the musical parameters driving the generation of music).
- it selects an action $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state s_t (here, the actions correspond to an increase or decrease of the musical parameter value)
- the action is performed and a time step later the agent receives a reward $r_{t+1} \in R$ and reaches a new state s_{t+1} (the reward is given by the listener's perception of musical tension)
- at time t the policy is a mapping $\pi_t(s,a)$ defined as the probability that $a_t = a$ if $s_t = s$ and the agent updates its policy as a result of experience

3.2 Returns

The agent acts upon the environment following some policy π. The change in the environment introduced by the agent's actions is communicated via the reinforcement signal r. The goal of the agent is to maximize the reward it receives in the long run. The discounted return R_t is defined as:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (1)$$

[2] http://puredata.info/Members/thomas/flext/

where $0 \leq \gamma \leq 1$ is the discount rate that determines the present value of future rewards. If $\gamma = 0$, the agent only maximizes immediate rewards. In other words, γ defines the importance of future rewards for an action (increasing or decreasing a specific musical parameter).

3.3 Value functions

Value functions of states or state-action pairs are functions that estimate how good (in terms of future rewards) it is for an agent to be in a given state (or to perform a given action in a given state).

$V^\pi(s)$ is the state-value function for policy π. It gives the value of a state s under a policy π, or the expected return when starting in s and following π. For MDPs we have:

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t | s_t = s\} \\ &= E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \end{aligned}$$

$Q^\pi(s,a)$, or action-value function for policy π, gives the value of taking action a in a state s under a policy π.

$$\begin{aligned} Q^\pi(s,a) &= E_\pi\{R_t | s_t = s, a_t = a\} \\ &= E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\} \end{aligned}$$

We define as optimal policies the ones that give higher expected return than all the others. Thus,$V^*(s) = max_\pi V^\pi(s)$, and $Q^*(s,a) = max_\pi Q^\pi(s,a)$ which gives $Q^*(s,a) = E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$

3.4 Value function estimation

3.4.1 Temporal Difference (TD) prediction

Several methods can be used to evaluate the value functions. We chose TD learning methods over Monte Carlo methods as they allow for online incremental learning. With Monte Carlo methods, one must wait until the end of an episode whereas with TD, one need to wait only one time step. The TD learning update rule for V^* the estimate of $V$is given by:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

where α is the step-size parameter or learning rate. It controls how fast the algorithm will adapt.

3.4.2 Sarsa TD control

For the transitions from state-action pairs we use a method similar to TD learning called sarsa on-policy control. On-policy methods try to improve the policy that is used to make decision. The update rule is given by:

$$\begin{aligned} Q(s_t,a_t) &\leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \\ &... \quad \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \end{aligned}$$

3.4.3 Memory: Eligibility traces (Sarsa(λ))

An eligibility trace is a temporary memory of the occurrence of an event.

We define $e_t(s,a)$ the trace of the state-action pair s, a at time t. At each step, the traces for all states decay by $\gamma\lambda$ and the eligibility trace for the state visited is incremented. λ represent the trace decay. It acts as a memory and sets the exponential decay of a reward based on previous context.

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & for\, s = s_t\,,\, a = a_t \\ \gamma\lambda e_{t-1}(s,a) & if\, s \neq s_t \end{cases}$$

we have the update rule

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a)$$

where

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

3.4.4 Action-value methods

For the action-value method, we chose a ϵ-greedy policy. Most of the time it chooses an action that has maximal estimated action value but with probability ϵ it instead select an action at random [9].

4. MUSICAL TENSION AS A REWARD FUNCTION

We chose to base the autonomous modulation of the musical parameters on the perception of tension. It has often been said that musical experience may be characterized by an ebb and flow of tension that gives rise to emotional responses [24, 25]. Tension is considered a global attribute of music, and there are many musical factors that can contribute to tension such as pitch range, sound level dynamics, note density, harmonic relations, implicit expectations, ...

The validity and properties of this concept in music have been investigated in various psychological studies. In particular, it has been shown that behavioral judgements of tension are intuitive and consistent across participants [7, 8]. Tension has also been found to correlate with the judgement of the amount of emotion of a musical piece and relates to changes in physiology (electrodermal activity, heart-rate, respiration) [26].

Since tension is a well-studied one-dimensional parameter representative of a higher-dimensional affective musical experience, it makes a good candidate for the one-dimensional reinforcer signal of our learning agent.

5. PILOT EXPERIMENT

As a first proof of concept, we looked at the real-time behaviour of the adaptive music system when responding to the musical tension (reward) provided by a human listener. The tension was measured by a slider GUI controlled by a standard computer mouse. The value of the slider was sampled every 100 ms. The listener was given the following instructions before performing the task: "use the slider to express the tension you experience during the musical performance. Move the slider upwards when tension increases and downward when it decreases".

The music generation is based on the base material described in section 2. The first monophonic voice controlled the right hand of a piano, the second monophonic voice an upright acoustic bass and the polyphonic voice the left hand of a piano. All the instruments were taken from the EXS 24 sampler from Logic Pro (Apple).

The modulation parameter space is of dimension 3. Dynamics modulation is obtained via a midi velocity gain factor between [0.0, 2.0]. Articulation is defined on the interval [0.0, 2.0] (where a value > 1 corresponds to a legato and < 1 a staccato). Tempo is modulated from 10 to 200 BPM. Each dimension was discretized into 8 levels, so each action of the reinforcement algorithm produces an audible difference. The reward values are discretized into three values representing musical tension levels (low=0, medium=1 and high=2).

We empirically setup the sarsa(λ) parameters, to $\varepsilon = 0.4, \lambda = 0.8, \gamma = 0.1, \alpha = 0.05$ in order to have an interesting musical balance between explorative and exploitative behaviors and some influence of memory on learning. ϵ is the probability of taking a random action. λ is the exponential decay of reward (the higher λ, the less the agent remembers). α is the learning rate (if α is high, the agent learns faster but can lead to suboptimal solutions).

5.0.5 One dimension: independant adaptive modulation of Dynamics, Articulation and Tempo

As our first test case we looked at the learning of one parameter at a time. For dynamics, we found a significant correlation ($r = 0.9, p < 0.01$): the tension increased when velocity increased (Figure 4). This result is consistent with previous psychological literature on tension and musical form [27]. Similar trends were found for articulation ($r = 0.25, p < 0.01$) (Figure 5) and tempo ($r = 0.64, p < 0.01$) (Figure 6). Whereas litterature on tempo supports this trend [28, 2], reports on articulation are more ambiguous [2].

5.0.6 Two dimensions: modulation of Tempo and Dynamics

When testing the algorithm on the 2-dimensional parameter space of Tempo and Dynamics, the convergence is slower. For our example trial, an average reward of medium tension (value of 1) is only achieved after 16 minutes of training (1000 s.) (Figure 7) compared to 3 minutes (200 s.) for dynamics only (Figure 4). We observe significant correlations between tempo ($r = 0.9, p < 0.01$), dynamics ($r = 0.9, p < 0.01$) and reward in this example, so the method remains useful for the study the relationship between parameters and musical tension. Nevertheless, in this setup, the time taken to converge towards a maximum mean reward would be too long for real-world applications such as mood induction or music therapy.

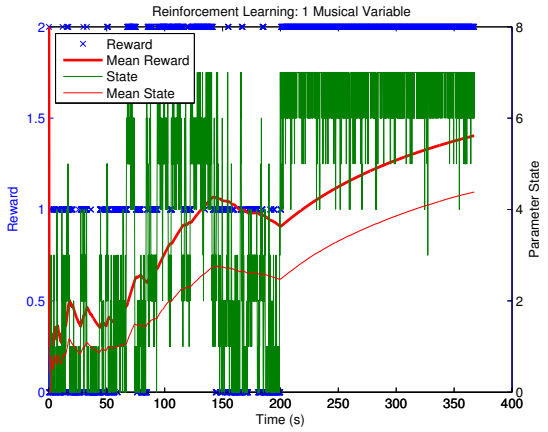


Figure 4. The RL agent automatically learns to map an increase of perceived tension, provided by the listener as a reward signal, to an increase of the dynamics gain. Dynamics gain level is in green, cumulated mean level is in red/thin, reward is in blue/crossed and cumulated mean reward is in red/thick.

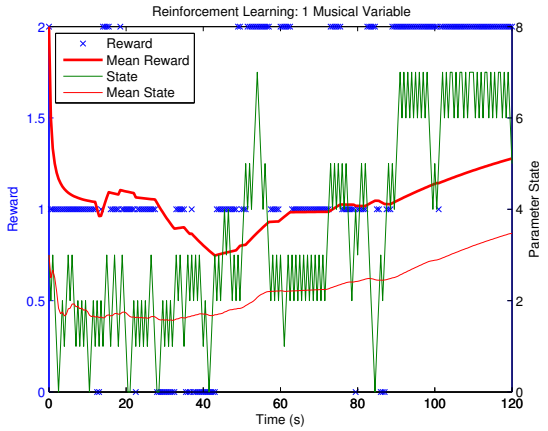


Figure 5. The RL agent learns to map an increase of perceive tension (reward) to longer articulations.

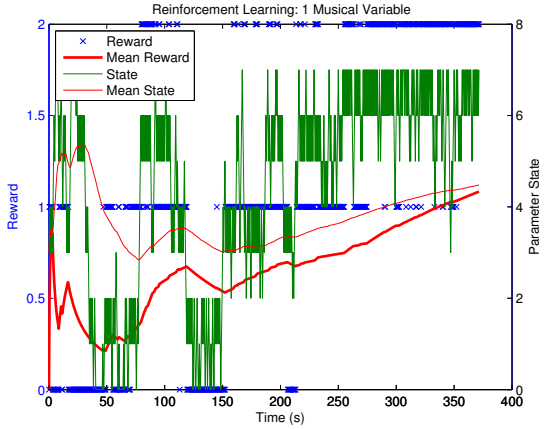


Figure 6. The RL agent learns to map an increase of musical tension (reward) to faster tempi.

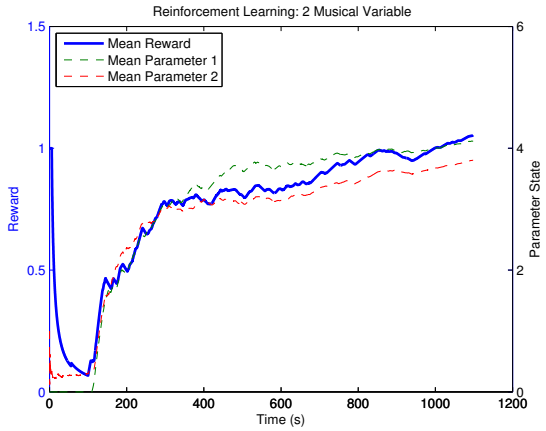


Figure 7. The RL agent learns to map an increase of musical tension (reward in blue/thick) to faster tempi (parameter 1 in green/dashed) and higher dynamics (parameter 2 in red/dashed).

5.0.7 Three dimensions: adaptive modulation of Volume, Tempo and Articulation

When generalizing to three musical parameters (three dimensional state space), the results were less obvious within a comparable interactive session time frame. After a training of 15 minutes, the different parameters values were still fluctuating, although we could extract some trends from the data. It appeared that velocity and tempo were increased for higher tension, but the influence of the articulation parameter was not always clear. In figure 8 we show some excerpt where a clear relationship between musical parameter modulation and tension could be observed. The piano roll representative of a moment where the user perceived low tension (center) exhibits sparse rhythmic density due to lower tempi , long notes (long articulation) and low velocity (high velocity is represented as red) whereas a passage where the listener perceived high tension (right) exhibits denser, sharper and louder notes. The left figure representing an early stage of the reinforcement learning (beginning of the session) does not seem to exhibit any special characteristics (we can observe both sharp and long articulation. e.g. the low voice (register C1 to C2) is not very dense compared to the other voices). From these trends, we can hypothesize that perception of low tension would relate to sparse density, long articulation and low dynamics which corresponds to both intuition and previous offline systematic studies [27].

These preliminary tests are encouraging and suggest that a reinforcement learning framework can be used to teach an interactive music system (with no prior musical mappings) how to adapt to the perception of the listener. To assess the viability of this model, we plan more extensive experiments in future studies.

6. CONCLUSION

In this paper we proposed a new synthetic framework for the investigation of the relationship between musical pa-

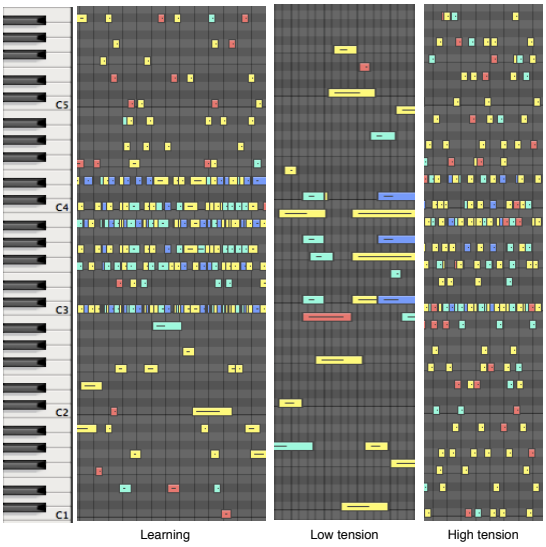


Figure 8. A piano roll representation of an interactive learning session at various stage of learning. At the beginning of the session (left), the musical output shows no specifc characteristics. After 10 min of learning, excerpts where low tension (center) and high tension reward is provided by the listener (right) exhibit different characteristics (cf text). The length of the notes correspond to articulation. Colors from blue to red correspond to low and high volume respectively.

rameters and the perception of musical tension. We created an original algorithmic music piece that can be modulated by parameters such as articulation, velocity and tempo, assumed to influence tension. The modulation of those parameters was autonomously learned in real-time by a reinforcement learning agent optimizing the reward signal based on the musical tension perceived by the listener. This real-time learning of musical parameters provides an interesting alternative to more traditional research on music and emotion. We could observe correlations between specific musical parameters and an increase of perceived musical tension. Nevertheless, one limitation of this method for real-time adaptive music is the time taken by the algorithm to converge towards a maximum average reward, especially if the parameter space is of higher dimensions. We will improve several aspects of the experiment in follow-up studies. The influence of the reinforcement learning parameters on the convergence needs to be tested in more details, and other relevant musical parameters will be taken into account. In the future we will also run experiments to assess the coherence and statistical significance of these results over a larger population.

7. REFERENCES

[1] L. B. Meyer, *Emotion and Meaning in Music*. The University of Chicago Press, 1956.

[2] A. Gabrielsson and E. Lindström, *Music and Emotion - Theory and Research*, ch. The Influence of Musical Structure on Emotional Expression. Series in Affective Science, New York: Oxford University Press, 2001.

[3] S. Le Groux, A. Valjamae, J. Manzolli, and P. F. M. J. Verschure, "Implicit physiological interaction for the generation of affective music," in *Proceedings of the International Computer Music Conference*, (Belfast, UK), Queens University Belfast, August 2008.

[4] C. Krumhansl, "An exploratory study of musical emotions and psychophysiology," *Canadian journal of experimental psychology*, vol. 51, no. 4, pp. 336–353, 1997.

[5] M. M. Bradley and P. J. Lang, "Affective reactions to acoustic stimuli.," *Psychophysiology*, vol. 37, pp. 204–215, March 2000.

[6] S. Le Groux and P. F. M. J. Verschure, "Emotional responses to the perceptual dimensions of timbre: A pilot study using physically inspired sound synthesis," in *Proceedings of the 7th International Symposium on Computer Music Modeling*, (Malaga, Spain), June 2010.

[7] W. Fredrickson, "Perception of tension in music: Musicians versus nonmusicians," *Journal of Music Therapy*, vol. 37, no. 1, pp. 40–50, 2000.

[8] C. Krumhansl, "A perceptual analysis of Mozart's Piano Sonata K. 282: Segmentation, tension, and musical ideas," *Music Perception*, vol. 13, pp. 401–432, 1996.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[10] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, "OMax brothers: a dynamic yopology of agents for improvization learning," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, p. 132, ACM, 2006.

[11] J. Franklin and V. Manfredi, "Nonlinear credit assignment for musical sequences," in *Second international workshop on Intelligent systems design and application*, pp. 245–250, Citeseer, 2002.

[12] B. Thom, "BoB: an interactive improvisational music companion," in *Proceedings of the fourth international conference on Autonomous agents*, pp. 309–316, ACM, 2000.

[13] N. Collins, "Reinforcement learning for live musical agents," in *Proceedings of the International Computer Music Conference*, (Belfast), 2008.

[14] A. Friberg, R. Bresin, and J. Sundberg, "Overview of the kth rule system for musical performance," *Advances in Cognitive Psychology, Special Issue on Music Performance*, vol. 2, no. 2-3, pp. 145–161, 2006.

[15] A. Friberg, "pdm: An expressive sequencer with real-time control of the kth music-performance rules," *Comput. Music J.*, vol. 30, no. 1, pp. 37–48, 2006.

ÕDAIKO:

A REAL TIME SCORE GENERATOR BASED ON RHYTHM

Filipe Lopes

Escola Superior de Música e das Artes do Espectáculo
(ESMAE-IPP) - Porto, Portugal
random@filipelopes.net

ABSTRACT

The interaction between composers and performers has recently acquired new challenges with the advent of scores in real time. Such systems potentiate new approaches to composition and performance by imposing new possibilities and constraints.

Õdaiko is a real-time graphical score generator and features a composer playing live electronic music, an assistant to the composer generating the scores and finally the performer(s). In this paper, I present Õdaiko, focusing on its implementations and the related composer-assistant-performer interactions as a basis for development.

1. INTRODUCTION

Recent computation and network power make possible to generate scores in real time thus opening a new paradigm for the composer-performer interaction.

In Õdaiko, I create a system relating rhythm, density changes over time, real time score generation and live electronic music. Scores generated in real-time pose novel approaches to composition and performance. The motivation to design Õdaiko was to explore the possibilities that such a system can contribute to the composer-performer relationship and to composition itself.

In Õdaiko, a score is generated in real time using graphic notation, providing cues to a performer (or group of performers) on when to play musical events. These musical events are pitches or other sounds relating to pre-defined electronic music events that are played live by the composer. In a performance using Õdaiko, the musician(s) have to respond to the music being played according to the temporal cues being generated by the system in real time.

2. STATE OF THE ART

Computers and network technologies have fostered and simplified the development of new interfaces while enhancing new relationships between musicians and composers.

The work done in this area has been increasingly growing in the last decade. Systems such as Automatic Notation Generators (ANG) [1], eScore [2] or Flock [3] address in different ways the use of real-time generated scores in performance, each of them focusing on different elements of interaction and different performance situations.

The ANG consists on various custom software tools developed to aid algorithmic composition. One of them is the LiveScore, developed by Harris Wulfson [1], which connects computers through a wireless network and displays a proportional notation system using a standard 5-line staff. A stochastically oriented algorithm accomplishes the data for the score. In this system synchronization is important and it is achieved by a moving vertical line indicating when and which to play. eScore was developed at Queens University in Belfast by McClelland and Alcorn [2]. Through an interactive drawing table, the composer can send musical gestures to the screen of the performers, and at the same time (s)he is triggering electronic music effects and/or samples. It features three different notation possibilities including page mode, scattering mode and scrolling mode, all of them based on standard music notation.

Flock [3] is an environment developed by Jason Freeman and Mark Godfrey that includes musicians, dancers, video, electronic sound and audience. Through a camera installed on the ceiling, the software keeps track of position of performers and generates musical notation according to the xy coordinates of the performers . The notation used is either standard or graphical. Throughout the performance, different musical density textures are created.

3. ÕDAIKO

Õdaiko continues the trend of live score generation. The main feature that distinguishes this system from the ones

presented above is its focus on rhythm and temporal-domain data. The sieves, developed by Xenakis [4], is the fundamental core of rhythm generation and music development, providing a way to place musical events in the time domain which the performers have to respond. All the previous examples presented here are pitch based whereas Õdaiko uses rhythm as the foremost structural element, not presenting any pitch indications.

3.1. Overview

Õdaiko offers flexible scoring possibilities. Based on a sieve formula, rhythmic events are positioned in time, allowing a composer to arrange rhythm and to modulate it in real time.

The Õdaiko environment produces graphical scores by the action of a human assistant who makes use of a dedicated interface to perform a pre-composed guidance score. In addition, the composer performs live electronic music having previously composed a guidance score for the assistant. The purpose of Õdaiko is to make available the possibility to structure music by means of rhythm and also offer the possibility to shape density and, consequently, formal structure in real time.

Õdaiko is a cross-platform, stand-alone application for performance developed in MaxMSP [5] and Processing [6]. The MaxMSP part of the application enables the composer to play live-electronic music, provides the interface for the human assistant, and sends data to be rendered as real-time scores by an application developed in Processing. All of the software modules communicate with themselves by Open Sound Control (OSC) [7].

3.2. The importance of sieves

Sieve theory was developed by Xenakis [4] and recently summarized by Ariza [8]. Xenakis used the sieves to produce pitch scales and rhythmic series in many of his compositions, although it could be used to generate data for several musical parameters. The sieve consists on a formula of one or more residual classes, combined or not by logic operators, producing a sequence of integers. A residual class consists of two integer values, a modulus (*M*) and a shift (*S)*. While the modulus can be any positive integer, the shift must be 0 or *M*-1. In Õdaiko, the modulus and the shift are notated *M@I,* as proposed by Ariza [8]. Given a sequence 4@0, each value in the sequence consists of all integers x where the following condition is satisfied: x%4==0. This is will produce the following sequence […-8, -4, 0, 4, 8…]. In Õdaiko only the positive part of the resulting integer sequence is used, up to a maximum of 300, to place events in time. It follows that if only one sieve is used then the sequence will be periodic with a period identical to the modulus. In Õdaiko, the sequence produced by the sieve is the reference to place musical events in a score. Although Xenakis used several combinations of sieves in his pieces, producing complex sequences, in Õdaiko only combinations of two can be used facilitating the assistant to make an accurate guess on how much a performer will play.

The sieve is one of the main elements in Õdaiko since it allows the assistant to work rhythm on the immediate present but also prepare the future shaping the scores in real time.

3.3. Beyond pitch notation paradigm

The focus of Õdaiko is the possibility to shape rhythm and form in real time upon an established framework, letting historical significant aspects like pitch, dynamics or articulations to happen spontaneously. Õdaiko notation doesn't provide pitch information, only rhythm.

Pitch, and the intrinsic musical features mention above, are dealt in rehearsals through collaboration between performers, assistant and composer, leading to the establishment of sectional areas and general sound textures. An established framework, or sectional area, is embedded in an assumption of materials fostering the composer to compose them in real time. This established sectional areas are the ones that the composer will use to create the guidance score, indispensable to the assistant.

For Õdaiko, I propose that rhythm should be the foremost feature of organization and development of music along a performance.

3.4. Typical Setup

A typical setup of Õdaiko will include a laptop for each performer, a laptop for the assistant and a laptop for the composer/manipulator of the electronic music. Each of these stations has dedicated software.

All the communications are accomplished using OSC [7] in a hierarchical structure from composer-assistant-performer. The composer controls the main clock and sends data to the assistant about succeeding sections of the piece, expressed in the form of letters, which he/she has to perform according to the pre-composed guidance score. The assistant then interprets and sends the data for the performer(s) computer(s) screen in order to render the score(s). In addition, the assistant can also alert the composer if he/she is doing, or is going to do, a solo.

3.5. The role of the assistant

In Õdaiko the composer performing electronic music is considerer as autonomous as each musician performing his instrument, creating the necessity to have someone dedicated for the score generation.

The assistant is someone who knows the system and essentially is able to facilitate the composer's musical ideas while generating the score(s).

The assistant is responsible for leading the music through the sectional areas, operating gradual passages in the scores, making his contribution active and vital in the musical outcome. Although he/she is interpreting a guidance score pre-composed by the composer (see Figure 1.), his interpretation and musical time flow contributes decisively to the musical outcome of the piece, making its influence significant in both the creative process of composition and performance.

Part A
Everybody start at the same time, with the same sieve, combined by the logical operation Union.
30@0
20@1
prob. size: big; shape: square

Part B
Slowly, instrument by instrument, start to break the pattern with regular patterns for each one.
Gongs should have sieves no less than 20@0

Part C
When all the instrument are playing different patterns, maintain the flow a little bit
Change to solo panel Sarons and one of the Bonangs – originally PR, MM, AP and JQ

Part D
Silence everybody at the same time
Electronic Music alone

Part E
After Electronic Music solo, one of the bonangs starts solo again – originally JQ

Part F
Send the same sieve to everybody (for instance 20@0)

Part G
Put in silence one instrument after the other but in such a way that the Gong finishes last.

Figure 1. Pre-composed score

4. NOTATION AND INTERFACE

Non-traditional scores offer demanding commitment from musicians, especially if they are generated in real time. Õdaiko uses a graphical approach to notation based on the classical piano-roll.

Each score can have three different situations. These can be the sieves panel, the solo panel or the silence panel. The passages between each panel are made through a cue on the top-right corner of the screen that fades out.

4.1. The Sieves panel

Each sieve places moving events (from right to left) on a graphical staff (see Figure 2).

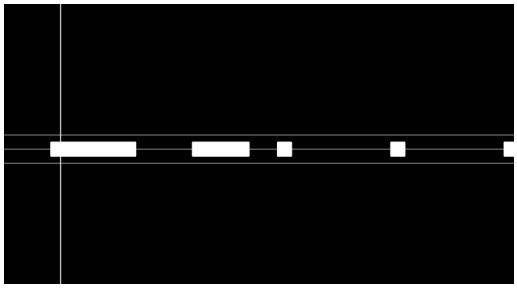


Figure 2. The sieves panel

These events should be performed when they hit a barline. The events can have five possible durations and three different shapes (triangle, square, or inverted triangle – see Figure 3). Each shape has a different performance intention. When the performer sees a square she/he should play the musical event with little or no modification over time whereas the triangle results in a dramatic change on the event. For example, if a square appears and the performer plays a chord, it should maintain itself still while if a triangle appears the same chord could be repeated over its duration while doing a crescendo or an accelerando. Although the shape for a stream of events is always the same, their durations is chosen making use of a probability scheme which includes the possibility of having most of the events big, most of them small or random size.

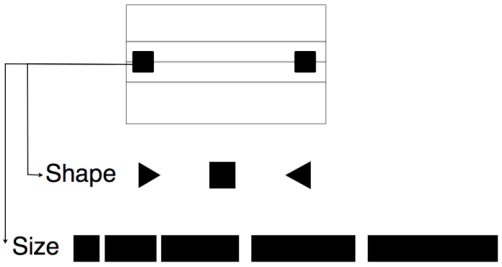


Figure 3. Event attributes

The score display is 1080 pixels width and 680 pixels height. The vertical barline default position is 120 pixels from the leftmost of the score layout. The smallest event is a square of 30 pixels and it represents the 16th note in reference to the main time/bpm clock. (see Figure 4). This means that there are 32 units divisions on the score. In the case of a sieve of 1@0 [0,1,2,3,4…300] it means that there will be an event in each unit division. In the case of 10@0 [0,10,20,30…300] it means that for each 10 unit divisions, it will be placed an event.

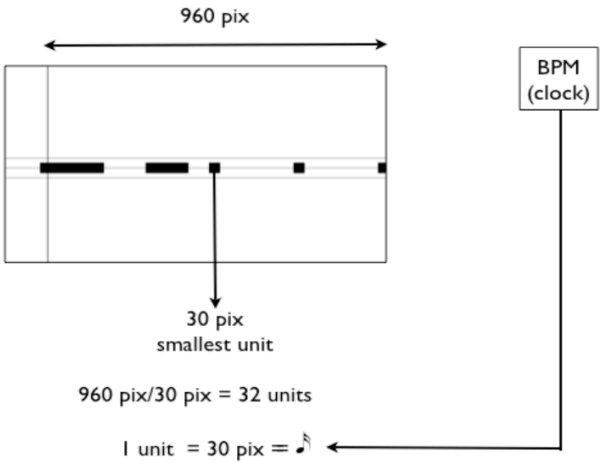


Figure 4. Pixel relationship

The assistant, in addition, can move the vertical barline along the x-axis. This results in smaller or bigger time spaces between the appearance of the event and its performance. The maximum displacement the vertical barline can have is up to 60 pixels from the right end of the score layout.

4.2. Solo panel

The solo panel (see Figure 5) allows the performer to do a solo. It features a line in the middle of the screen and a circumference in the middle of the line. Both randomly change brightness with occasional blobs of white shapes masking everything.

There is no specific musical meaning for the animation, however it has been helpful to enhance performers less comfortable with free improvisation, providing something to keep them going.

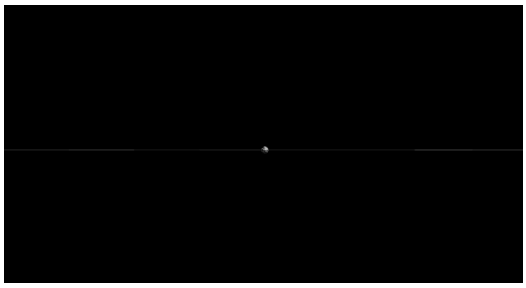


Figure 5. The solo panel

4.3. Silence panel

A black screen tells the performer(s) to stop playing

4.4. Assistant Interface

The application (see Figure 6) includes a section where the assistant can choose which panel to display in each performer screen, which sieve to use, the probability size of each event and its shape. He can also move the vertical barline that provides the cue of each performer. There is also a more general section that includes the possibility to change, at the same time, for the same panel, all the laptops. It is also possible to send the same sieve to all the laptops, enhancing synchronization of the ensemble, as well as sending a warning to the composer if he/she is doing a solo.

In addition, there are tools that provide the assistant to check if the wireless network is working properly and also to receive information, sent from the composer, about which section of the piece is being played.

4.5. Electronic Music Modules

Electronic music is performed using modules (see Figure 7). Featuring a maximum of 25 modules, they are designed in such a way to share common rhythmic characteristics.

Each module has two important parameters: The time distance between consecutive events and their associated time periodicity. The first one controls the time space for each event, and the second one, within the time space defined by the first one, controls its linked regular or irregular division of time. The composer chooses the time reference in beats per minute, which is the same that controls Õdaiko score engine.

A composer can sonically design each module, programming it in MaxMSP making use of a template. In this way and by the benefit of using a computer, the composer can address real-time synthesis based on rhythm at a sample level or control electronic music gestures making use of musical processes defined in each module.

In addition, the use of electronic music also offers the possibility to blend acoustical with electronic music.

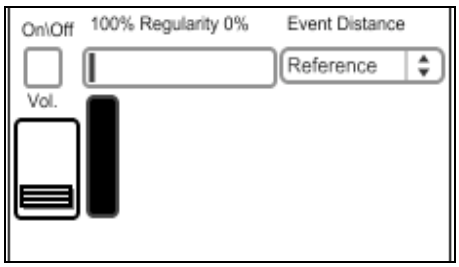


Figure 7. Electronic Music Module

Figure 6. Assistant Interface

5. COMPOSITION AND PERFORMANCE

Õdaiko allows the composer to shape rhythm and form in real time.

Using sieves allows the possibility of complex rhythmic textures, on a vertical or horizontal level. Aspects like beat, pulse and meter along with polyrhythmic and polymetric music can be enhanced within a group of performers. The sieves also allow exploring the aspect of time and form while generating and changing it in real time. Nevertheless, a long performance made only by the sieves panel could become troublesome and musically uninteresting. Since Õdaiko lends itself for performers comfortable with improvisation, a solo panel was added in order to free the performance from continuous limitations. It is also clear that, regarding rhythm, silence is as important as sound, thus, a panel of silence was incorporated. The mixture of all these three possibilities is very exciting and rich for composition and interactive systems.

The assistant is a vital part of the system since he is in charge of generating the scores and mediating the composer's thoughts. Upon a given structure, the assistant generates the scores and operates the gradual changes between each section of the piece, assuming him as a lead element for the musical outcome.

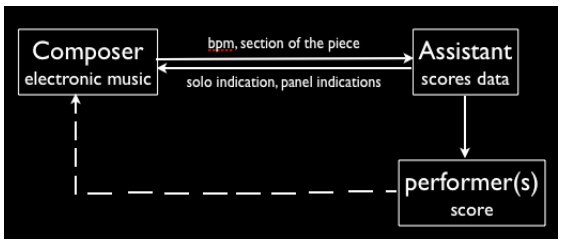


Figure 8. Performance Relationship Models

6. PERFORMANCE ISSUES

Õdaiko is tool for real time composition developed in a very personal compositional perspective. Generally speaking it allows a composer to address flexible musical densities in a stochastically way and in real time.

The most obvious challenge is the way the performers will interpret the graphical score giving the fact that no pitch indication is provided. This is particularly challenging as it goes against the traditional practice of instruments and also most of the academic music teaching. The performers also abandon the concepts of form and development since the scores are generated in real time. These features make the rehearsals one of the more important aspects in order to achieve a good performance.

The most musically effective performances have involved the use of percussion instruments such as the Gamelan or, in the case of pitch instruments, coherent pitch zones. This lends the performers a sonic quality that offers constancy in combination with recurrent electronic music pitched sounds samples.

7. FUTURE WORK

Performance based studies will be accomplished in order to attain a better insight on the matter of rhythm relations and graphical notation. A second parallel study will be about the constraints that a real time score has in performance.

Recent pieces composed for Õdaiko demonstrated that rhythm and real-time score generation can be compositionally rewarding and that the possibility of screen-based notation can be immense.

Different musicians and instruments, from traditional acoustical ones to electronic based ones, will be invited to play Õdaiko and share ideas and thoughts.

Õdaiko is intended for public release.

8. CONCLUSIONS

In this paper I've shown Õdaiko, a real time graphical score system based on rhythm that enables the composer to address rhythm in a structural manner. This way of addressing composition promotes the composer to establish rhythm relations "in the present" but also to focus on musical form, thus, the future.

The usage of real time graphical generated scores enhances new relations between composer, performers and performance itself. I then presented some practical outcome and future work to be developed.

9. ACKNOWLEDGEMENTS

Õdaiko was developed as part of my master's degree [9]. I would like to thank Joel Ryan and Paul Berg my advisors for all their support and guidance. I also would like to thank Carlos Guedes and Rui Penha for all their ideas and criticism.

10. REFERENCES

[1] Barret, Douglas G.,Winter, M., Wulfson, H. "Automatic Notation Generators". *In Proceedings of the 2007 New Interfaces For Musical Expression*, New York, EUA, 2007.

[2] McClelland, C., Alcorn, M. "Exporing New Composer/Performer Interactions Using Real-Time Notation*" In Proceedings of the International Computer Music Conference*, Belfast, Ireland, 2008.

[3] Freeman, J., Godfrey, M. Technology, "Real-Time Notation, and Audience Participation in Flock". *In International Computer Music Conference*, Belfast, Ireland, 2008.

[4] Xenakis, I. "Formalized Music, Thought and Mathematics in Music*" Pendagron Press*, 1992.

[5] (http://www.cycling74.com)

[6] (http://www.processing.org)

[7] (http://opensoundcontrol.org)

[8] Ariza, Christopher "The Xenakis Sieve as Object: A New Model and Complete Implementation*" In Computer Music Journal*, Summer 2005, Vol. 29, No.2, pag 40-60, MIT.

[9] Lopes, Filipe "Õdaiko a Real Time Score Generator" Master Thesis, Institute of Sonology, The Hague, Holland, 2009

STYLE EMULATION OF DRUM PATTERNS BY MEANS OF EVOLUTIONARY METHODS AND STATISTICAL ANALYSIS

Gilberto Bernardes
Faculdade de Engenharia da Universidade do Porto
bernardes7@gmail.com

Caros Guedes
INESC, Porto
carlosguedes@mac.com

Bruce Pennycook
University of Texas, Austin
bpennycook@mail.utexas.edu

ABSTRACT

In this paper we present an application using an evolutionary algorithm for real-time generation of polyphonic drum loops in a particular style. The population of rhythms is derived from analysis of MIDI drum loops, which profile each style for subsequent automatic generation of rhythmic patterns that evolve over time through genetic algorithm operators and user input data.

1. INTRODUCTION

Application kin.genalgorthythm is a Pd patch that uses a genetic algorithm (GA) and statistical analysis data gathered from pre-existing MIDI drum loops in order to perform automatic variations on existing drumming styles. This is done in real time and is mainly intended for real-time performance driven by user-controlled data.

GAs are known for being a powerful technique for problem solving by searching in a vast space of possibilities, created from conventional genetic operators such as crossover and mutation, by means of a fitness function, which typically consists of an objective function that is able to rank all chromosomes in order to find an optimal solution. This method has been widely used as a creative tool in music, particularly for the development of variation of music sequences [1,2,3]. However, as noted by Biles [1], encoding a fitness function in a GA for real-time operation in music is a major issue due to the complexity of the aesthetic judgments related to this task.

Papadopoulos and Wiggins [5] present a categorization of musically-oriented GA applications based on the use of fitness functions. They establish a major difference between the use of an objective fitness function and a human one. In other words, they discuss the difference between evaluations of chromosomes based on formally stated computable functions, or in human judgment as a means of replacing the fitness function.

The solution we adopt here does not use a fitness function, and encodes several musical constraints directly in the GA's operators involved in the generation of new populations. This is done to avoid non-musical search spaces.

Our approach is inspired on Arne Eigenfeldt's *Kinetic Engine* [2,4] and focuses on two points: (1) an elaboration on the crossover technique proposed by Eigenfeldt; (2) the introduction of a metrical-supervision procedure on the mutation operator. Our elaboration on the crossover technique proposed by Eigenfeldt considers different lengths of the parental chromosome. The metrical supervision procedure operated on the mutation operator mutates the events according to the meter of the drum loop by using the metric indispensability principle presented by Clarence Barlow [6].

The output of the algorithm thus results from the selection of the best candidate from an evolving population of metrically coherent rhythms produced by the GA. Subsequently, the user can further control the musical output by introducing two parameters – density and complexity of events. These parameters can be introduced in real or non-real time, but they do not affect the metrical coherence or the style of the rhythmic sequence.

2. SYSTEM DESIGN

Below we present the design scheme for kin.genalgorhythm. The two blocks at the left (Analysis and Stored analysis) deal with previous analysis and storage of MIDI drum loops. The data gather and stored from these blocks will feed and initialize the main block (Generation of metrically supervised population) that uses a genetic algorithm containing two operators – crossover and mutation. This central block is responsible for the creation of a finite number of metrically coherent populations. Finally, at the right we have two blocks (User input and Performance), which deal directly with the performative aspects of the algorithm. These blocks have two roles. The first (User input) is to find the optimal solution from the generated offspring population, based on user input data – complexity and density. The second (Performance) appropriately distributes the rhythmic phrase among several drum parts.

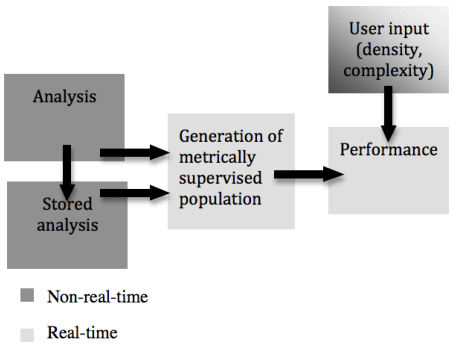
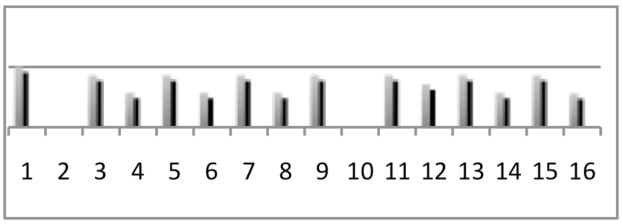


Figure 1. The design scheme for kin.genalgorithm.

3. INITIALIZATION

We do previous non-real-time analyses of polyphonic MIDI drum loops in a particular style in order to inform the algorithm about the specificities of that style. The analysis block of the algorithm will output the probability distribution of event occurrences for each individual part within a measure. In other words, each resulting vector corresponds to the normalized histogram of events that occur in each subdivision of a pre-defined measure.
A general vector, which takes into account all instrumental parts is also defined in order to gather the metrical accent distribution of the style under analysis. We use two different collections of MIDI loops containing style and tempo annotations available in the Free Pack from Groove Monkey [7] and Apple's Logic Pro [8]. In Figure 2 we can observe the normalized probability distribution for the Mambo loops from the Groove Monkey Pack [7].



1.00	0.00	0.86	0.0.57	0.86	0.57	0.86	0.57	0.86	0.00	0.86	0.71	0.86	0.57	0.86	0.57

Figure 2. Normalized probability distribution obtained with the analysis of the Mambo loops from the Groove Monkey Pack [7] of the 16 pulses comprising the 16th note level of a 4/4 meter.

Prior to the analysis, all the loops labeled with the same style (e.g. "70s Street Drums") are quantized, assembled together in a MIDI sequencer, and subsequently exported as a sole MIDI file. Our application uses a feature that can store the analyses and provide an instant access for future use.

The analysis block is dependent on three variables that have to be introduced by the user: (1) the tempo of the loop, indicated in beats per minute (BPM); (2) the length of the chromosome to be analyzed, which corresponds to the number of subdivisions in each measure (for example in a 4/4 meter at the 16th level, we would have 16 subdivisions); and (3) the amount of subdivisions each pulse contains (drawing on the previous example, and assuming that we have 16 subdivision, we would have to inform the algorithm that each pulse contains 4 subdivisions, in order to define a 4/4 meter at the 16th note level).

4. EVOLUTIONARY METHODS

The evolution of the GA is mainly governed by two principles: crossover and mutation. In this section we present the transformations we operated in principles advanced by Eigenfeldt [2,4] as well as a novel model to metrically supervise the mutation operator.

4.1 Crossover

Crossover is a standard evolutionary technique in GA, in which portions of two individuals (parents) are spliced together at random split points and rejoined interchangeably in order produce a new variation (children). When combining the two parental chromosomes the idea behind this operation is that the resulting chromosomes may be better than both of the parents if it takes the best characteristics from each of the parents.

As Eigenfeldt points [2,4], crossover is not a usual developmental technique in music due to the arbitrary method used to determine the splitting point. In order to circumvent this problem, Eigenfeldt proposes the use of a single parent chromosome in which a first-order Markov chain is applied to perform the crossover. However, since most styles analyzed here do not present much variation, we found a tendency to obtain almost the exact same sequence in all offspring populations. Several experiments showed us that if we increase the dimension of the parental chromosome, the amount of novelty increases as well. Therefore, an average length 60 beats (around 30 seconds of a sequence at 120 BPM) revealed much more proficient results.

The resulting chain of events is based on a transition table, which gives the probability of moving from one state to another. The pairs of successive rhythmic cells are coupled using the object **anal** that computes the transition matrix. Object **prob** is then used to generate the events according to the transition matrix. Both objects belong to the cyclone external library in Pd-extended [9]. In order to compute the transition matrix, we convert the binary representation of each rhythmic cell of the offspring chromosomes sequences to decimal (see Figure 3). When making the analysis, the user already defines the length of each rhythmic cell. Our method ignores the duration of each note, as this does not affect the outcome – we are dealing with percussive instruments with a natural decay. Special attention is paid to beginnings of offspring phrases, which are restricted to the first rhythmic cell of the analyzed material.

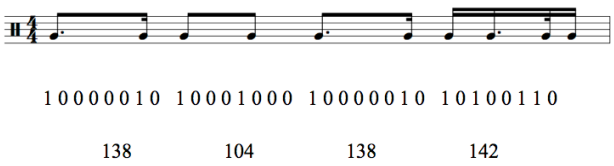


Figure 3. Example of an offspring in musical, binary, and decimal representations.

4.2 Mutation

The purpose of mutation is to introduce and preserve diversity in the offspring population in order to access a wider range of possible musical densities. A common usage of this technique, involves altering a certain number of arbitrary bits in a genetic sequence depending on the level of desired mutation.

The method we use is a variation of a stochastic algorithm, usually known as roulette-wheel selection (RWS). For each bin, a random number is generated and compared against a segment division, deciding if the subdivision in cause should be mutated or not.

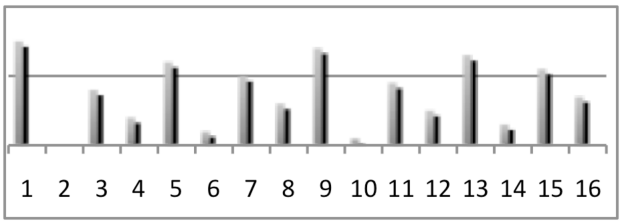
In our approach, we bias the mutation operator in order to produce a metrically coherent output. This is

done using a decision-making process that takes into account Barlow's metrical indispensability principle [6]. The amount of mutation is assigned by default to the level of desired density as input by the user. Low and high densities are assigned to low and high mutation ratios respectively. The metrical coherence of the offsprings is preserved independently from the mutation ratio.

4.2.1 Metrically-supervised Mutation

Barlow's metric indispensability principle defines the probabilistic weight each accent at metrical level on a given meter should have in order for that meter to be perceived – i.e. how *indispensible* is each accent at a certain metrical level for a meter to be felt.

The accents' weights are calculated by a formula that takes into account the meter (e.g. 4/4) and the metrical level (e.g. 16th note) for which one wants to calculate the indispensabilities. The metrical level is defined by a unique product of prime factors which equals the number of pulses at that metrical level, and takes into account the division (binary or ternary) at higher levels. For example, the six pulses comprising the 8th-note level in a ¾ meter would be defined as 3x2 (representing the *three* quarter notes at the quarter-note level that subdivide into *two* 8th-notes at the level below), whereas the six pulses comprising the 8th-note level in 6/8 would be represented as 2x3 (*two* dotted quarters that subdivide into *three* 8th-notes). Below we show the normalized distribution for the 16 pulses comprising the 16th note level in 4/4.



1.00	0.00	0.53	0.27	0.80	0.13	0.67	0.40	0.93	0.07	0.60	0.30	0.87	0.02	0.73	0.47

Figure 4. Probability distribution given by Clarence Barlow's indispensability formula for the 16 pulses comprising the 16th note level of 4/4, which is defined as 2x2x2x2.

Barlow's metrical indispensability algorithm has been effectively used to automatically generate rhythm at a certain meter. In this application, we use these distributions as a metrical template that is applied to the mutation operator of the GA, by supplying the threshold mutation values for each of the measure subdivisions.

Aside from metrical indispensability, we implemented two other different vectors that can be used instead. One is the probability distribution vector of each analyzed style, and the other is the mean of the product of the last vector and the metric indispensability vector. Further research should explore other possible metrical templates by using other vectors. Empirically, we observe almost the same quality results with the current three possibilities.

5. PERFORMANCE

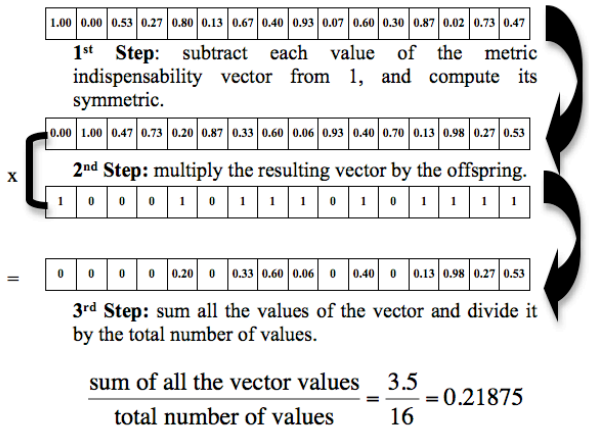
Initially, the user must define the style she or he wants to perform; either by selecting one of the pre-stored analyses or by analyzing a midi drums sequence. In order to create variation at the macro structural level the user can assign values for two parameters (density and complexity), either in real- or non-real time.

5.1 Density and complexity parameters

Whenever a genetic operator (crossover or mutation) is applied to create offspring populations, the values of density and complexity for each offspring are calculated.

The density of each rhythmic phrase corresponds to the rate of the attacks per measure. For example, if we have the following string: [1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1], the density is calculated by diving the sum of all the string values (10), and the total number of values of the string (16), which gives a density of 0.625.

The complexity parameter specifies the degree of syncopation by weighting the events occurring at the weaker pulses of the metric structure. It is calculated in three steps: 1- by multiplying each offspring vector by the symmetric of the indispensability vector around 0.5 (i.e. by subtracting each value from 1.0); 2- by summing all the values resulting from the operation; and 3- by dividing the sum by the number of elements in the vector. This way, vectors containing more events on weaker pulses of the metrical level (i.e. more syncopated) have higher complexity values. Figure 5 describes the procedure.



$$\frac{\text{sum of all the vector values}}{\text{total number of values}} = \frac{3.5}{16} = 0.21875$$

Figure 5. Steps for computing the complexity of each vector.

For each offspring population the program computes a table like the one shown below.

Sequence	Density	Complexity
10000010 10001000 10000010 10100110	0.3125	3.77419
10000010 10101110 10001010 10001010	0.40625	4.93548
10000000 10001000 10000000 10100010	0.21875	3

Table 1. Example of a density and complexity measurements.

5.2 Selection for Performance

The user must provide values corresponding to the density and complexity of the phrase for performance, so that the algorithm at runtime chooses offsprings that are closely related to the values input by the user.

The selection is done by finding the offspring that presents the highest correlation between a vector defined by the two values given by the user (density, complexity), and the correspondent vectors for all generated population. The density and complexity values can be assigned and stored previously in a table or altered in real-time. In Figure 6 we can observe the two different input approaches: the non-real time methods that can be stored in a timeline (graph on the top right of the window), and the real-time method by directly adapting the density and complexity sliders (below the timeline). The algorithm reads a new value on the beginning of each new sequence.

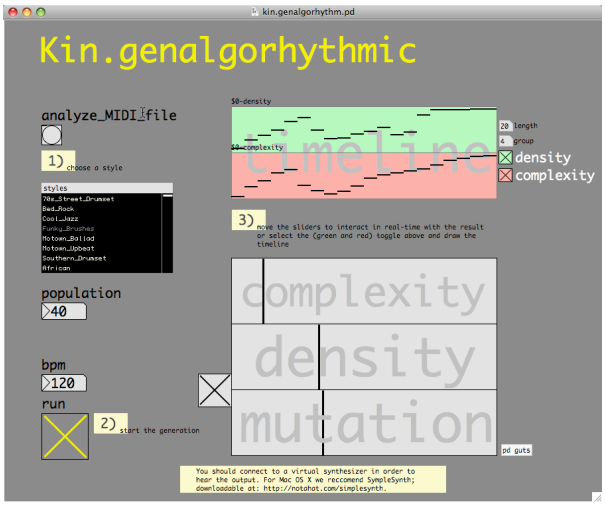


Figure 6. Main window in Pure Data presetting the interface where the user can both store the complexity and density values in a timeline (in the upper right) or alter the values in real-time directly on the sliders (below the timeline).

5.3 Performance

At runtime, the algorithm (1) will sequentially read the selected offspring, and (2) will evaluate in all probabilistic distribution tables (each corresponding to a different part: i.e. snare drum, hi-hat) to select if that part will play or not.

We felt the necessity to implement two rules that constrain the output in order to avoid certain non-musical situations. The first was to avoid the coincidence of similar instruments, and the second was to avoid the excessive appearance of phrase beginning accents such as crash-cymbal hits. In order to solve the first situation we restricted similar instruments to play in the same subdivision of the beat, opting for the strongest/sharpest one – e.g. if we have for the same beat a snare drum and a steel snare drum, we will only play the second one. For the second situation, we restricted the appearance of strong accents (such as crash cymbals) to the first of each four phrases.

A special attention should be paid to the tempo of the generative patterns, which by default is assigned to the tempo the user has introduced during the analysis. Certain styles have a clear BPM range associated with it, and this factor can determinant for the perception of a certain style. For example if we consider styles such as House music and Funk, the tempo difference is crucial to denote the difference between them, since they have almost the same generative profile.

6. CONCLUSION

In this paper we present an application that generates rhythmic patterns in a specific style by means of a genetic algorithm and statistical analysis. Major differences from similar software include the focus on the emulation of different styles and special attention is given to the metrical coherence of the output. The use of Barlow's indispensability algorithm [6] proved to be an efficient method for assuring metrical coherence in the offspring generation by mutation.

The software, analyses and generated samples used in the study are available at: http://sites.google.com/site/kineticproject09/home.

7. ACKNOWLEDGMENTS

This joint research project is made possible thanks to the support from the Portuguese Foundation for Science and Technology (grant UTAustin/CD/0052/2008) and the UT Austin | Portugal Program in Digital Media [10].

8. REFERENCES

[1] Biles, J. (1994). GenJam: A Genetic Algorithm for Generating Jazz Solos. *Proceedings of the International Computer Music Conference*

[2] Eigenfeldt, A. (2009). The Evolution of Evolutionary Software Intelligent Rhythm Generation in Kinetic Engine. *Proceedings of EvoMusArt 09, the European Conference on Evolutionary Computing*, Tübingen, Germany.

[3] Martins, J. and Miranda, E. (2007). Emergent rhythmic phrases in an A-Life environment. *Proceedings of ECAL workshop on Music and Artificial Life*, Lisbon, 2007.

[4] Eigenfeldt, A. (2006). Kinetic Engine: Toward an Intelligent Improvising Instrument. *Proceedings of Sound And Music Computing*, Marseille, France.

[5] Papadopoulos, G. and Wiggins, G. (1999). AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects," *Proceedings of the AISB'99 Symposium on Musical Creativity*, Edinburgh, UK.

[6] Barlow, C. (1987). Two essays on theory. *Computer Music Journal*, 11, 44-60.

[7] http://www.groovemonkee.com/home/

[8] http://www.apple.com/logicstudio/

[9] http://puredata.info/

[10] http://www.utaustinportugal.org/

SIMPLE TEMPO MODELS FOR REAL-TIME MUSIC TRACKING

Andreas Arzt
Department of Computational Perception
Johannes Kepler University Linz

Gerhard Widmer
Department of Computational Perception
Johannes Kepler University Linz
The Austrian Research Institute
for Artificial Intelligence (OFAI)

ABSTRACT

The paper describes a simple but effective method for incorporating automatically learned tempo models into real-time music tracking systems. In particular, instead of training our system with 'rehearsal data' by a particular performer, we provide it with many different interpretations of a given piece, possibly by many different performers. During the tracking process the system continuously recombines this information to come up with an accurate tempo hypothesis. We present this approach in the context of a real-time tracking system that is robust to almost arbitrary deviations from the score (e.g. omissions, forward and backward jumps, unexpected repetitions or re-starts) by the live performer.

1. INTRODUCTION

Real-time audio tracking systems, which listen to a musical performance through a microphone and automatically recognize at any time the current position in the musical score, even if the live performance varies in tempo and sound, promise to be useful in a wide range of applications. They can serve as a (musical) partner to the performer(s) by e.g. automatically accompanying them, interacting with them or supplementing their art by the creation of visualizations of their performance.

In this paper we propose a very simple and general method for incorporating learned tempo models into real-time music trackers. These tempo models need not reflect one specific way of how to perform a piece of music, but rather illustrate many different possible performance strategies (in terms of timing and tempo). We present this approach in the context of a real-time music tracking system that is extremely robust in the face of almost arbitrary structural changes (e.g. disruptions or re-starts) during a live performance.

This unique ability distinguishes our real-time tracking system from the two major advanced score followers that have been developed in recent years. These systems have two quite different domains in mind. While Christopher Raphael's *'Music Plus One'* [1] focuses on the automatic accompaniment of music containing a quite regular pulse, like western classical music, where close synchronization between the solo and the accompanying parts are required, Arshia Cont's system 'Antescofo' [2] addresses a slightly different domain, namely, contemporary music by composers like Boulez, Cage and Stockhausen, with musical characteristics quite different from 'classical' music. During the tracking process both systems are guided by sophisticated tempo models.

In contrast to the above-mentioned systems, which are based on probabilistic models, our music follower uses *Online Dynamic Time Warping (ODTW)* as its basic tracking algorithm (at multiple levels – see Section 3). Even without a predictive model of tempo, this algorithm is surprisingly robust. But for passages with extremely expressive timing, knowledge about plausible performance strategies is needed to improve the precision of real-time alignment. In this paper we will show two simple and very general ways of doing so, the second of which actually permits the system to adapt to different ways of playing without separate training each time.

In the following, we first re-capitulate the basic principles of our approach to on-line music following (Section 2), briefly point to a recent extension that makes the algorithm robust to almost arbitrary disruptions in a performance (Section 3; the details of this are described in a separate paper [3]), and then describe two simple, but effective ways of introducing expressive tempo information into the tracking process in Sections 4 and 5.

2. A HIGHLY ROBUST MUSIC TRACKER

Our approach to score following is via audio-to-audio alignment. That is, rather than trying to transcribe the incoming audio stream into discrete notes and align the transcription to the score, we first convert a MIDI version of the given score into a sound file by using a software synthesizer. The result is a 'machine-like', low-quality rendition of the piece, in which, due to the information stored in the MIDI file, we know the time of every event (e.g. note onsets).

2.1 Data Representation

The score audio stream and the live input stream to be aligned are represented as sequences of analysis frames, computed via a windowed FFT of the signal with a hamming window of size 46ms and a hop size of 20ms. The data is mapped into 84 frequency bins, spread linearly up

to 370Hz and logarithmically above, with semitone spacing. In order to emphasize note onsets, which are the most important indicators of musical timing, only the increase in energy in each bin relative to the previous frame is stored.

2.2 On-line Dynamic Time Warping (ODTW)

This algorithm is the core of our real-time audio tracking system. ODTW takes two time series describing the audio signals – one known completely beforehand (the score) and one coming in in real time (the live performance) –, computes an on-line alignment, and at any time returns the current position in the score. In the following we only give a short intuitive description of this algorithm, for further details we refer the reader to [4].

Dynamic Time Warping (DTW) is an off-line alignment method for two time series based on a local cost measure and an alignment cost matrix computed using dynamic programming, where each cell contains the costs of the optimal alignment up to this cell. After the matrix computation is completed the optimal alignment path is obtained by tracing the dynamic programming recursion backwards (*backward path*).

Originally proposed by Dixon in [4], the ODTW algorithm is based on the standard DTW algorithm, but has two important properties making it useable in real-time systems: the alignment is computed incrementally by always expanding the matrix into the direction (row or column) containing the minimal costs (*forward path)*, and it has linear time and space complexity, as only a fixed number of cells around the forward path is computed.

At any time during the alignment it is also possible to compute a *backward path* starting at the current position, producing an off-line alignment of the two time series which generally is much more accurate. This constantly updated, very accurate alignment of the last couple of seconds will be used heavily throughout this paper. See also Figure 1 for an illustration of the above-mentioned concepts.

Improvements to this algorithm, focusing both on adaptivity and robustness, were presented in [5] and are incorporated in our system, including the 'backward-forward strategy', which reconsiders past decisions (using the backward path) and tries to improve the precision of the current score position hypothesis.

In the following, we will give a short description of a dynamic and general solution to the problem of how to deal with structural changes effectively on-line, and then describe and evaluate our main new contribution: two ways to estimate the current tempo of a performance on-line, and how to use this information to improve the alignment.

3. 'ANY-TIME' REAL-TIME AUDIO TRACKING

In [3] we introduced a unique feature to this system, namely the ability to cope with arbitrary structural deviations from the score during a live performance. At the core is a process that continually updates and evaluates high-level hypotheses about possible current positions in the score, which are then verified or rejected by multiple instances of the basic alignment algorithm described above. To guide our

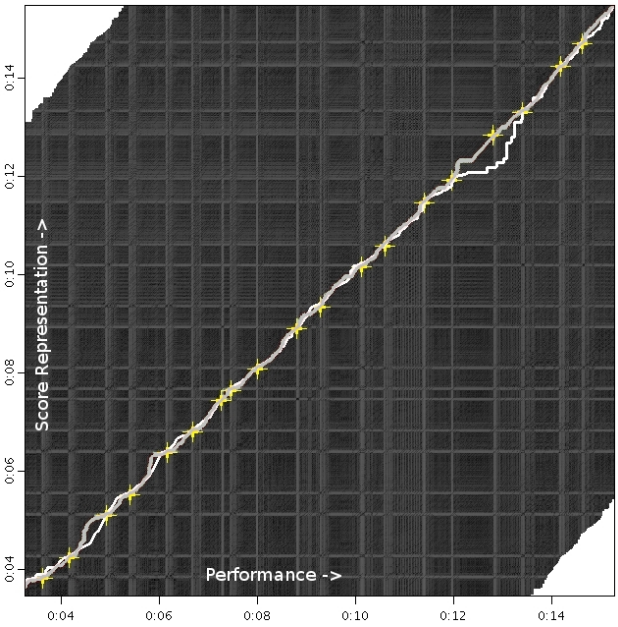


Figure 1. Illustration of the ODTW algorithm, showing the iteratively computed forward path (white), the much more accurate backward path (grey, also catching the one onset that the forward path misaligned), and the correct note onsets (yellow crosses, annotated beforehand). In the background the local alignment costs for all pairs of cells are displayed. Also note the white areas in the upper left and lower right corners, illustrating the constrained path computation around the forward path.

system in the face of possible repetitions and to avoid random jumps between identical parts in the score, we also introduced automatically computed information about the structure of the piece to be tracked. We chose to call our new approach *'Any-time Music Tracking'*, as the system is continuously ready to receive input and find out what the performers are doing, and where they are in the piece.

Figure 2 visually demonstrates the capabilities of our system. In this case 5 different performances of the Prelude in G minor Op. 23 No. 5 by Sergei Rachmaninoff are tracked that start not at the beginning, but 20 bars into the piece. While the basic system finds the correct position after a long timespan (basically by chance), our 'any-time' tracker almost instantly identifies the correct position.

While testing this real-time tracking system with complex piano music played with a lot of expressive freedom in terms of tempo changes, we realized the need for a tempo model to improve the alignment accuracy and the robustness of our system. In the following we propose two simple tempo models, one only based on the analysis of the most recent couple of seconds of the live performance (Section 4) and one having access to automatically extracted additional knowledge about possible future tempo developments (Section 5). The result will be a robust real-time tracker that is able to adapt to and even anticipate tempo changes of the performer, thus leading to a significant increase in alignment precision.

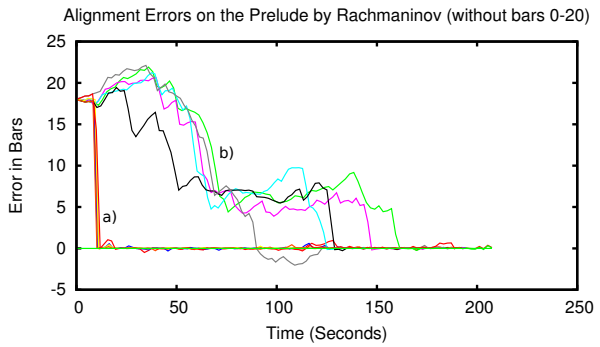


Figure 2. 'Starting in the middle': A visual comparison of the capabilities of the tracker in [5] and the 'any-time' real-time tracking system described in [3]. 5 performances of the g minor Prelude by Rachmaninoff, with bars 0-20 missing, are aligned to the score by both systems. For all performances, the 'any-time' real-time tracker (a) almost instantly identifies the correct position, while the old system (b) finds the correct position by mere chance.

4. A (VERY) SIMPLE TEMPO MODEL

4.1 Computation of the Current Tempo

The computation of the current tempo of the performance (relative to the score representation) is based on a constantly updated backward path starting in the current position of the forward calculation. As the backward path, in contrast to the forward path which has to make its decisions on-line, has perfect information about the performance – at least up to the current position in the performance –, it is much more accurate and reliable than the forward path (see also Figure 1).

Intuitively, the slope of such a backward path represents the relative tempo differences between the score representation and the actual performance. Given a perfect alignment, the slope between the last two onsets would give a very good estimation about the current tempo. But as the correctness of the alignment of these last onsets generally is quite uncertain, one has to discard the last few onsets and use a larger window over more note onsets to come up with a reliable tempo estimation.

In particular, our tempo computation algorithm uses a method described in [6]. It is based on a rectified version of the backward alignment path, where the path between note onsets is discarded and the onsets (known from the score representation) are instead linearly connected. In this way, possible instabilities of the alignment path between onsets (as, e.g., between the 2^{nd} and 3^{rd} onset in the lower left corner in Fig.1) are smoothed away.

After computing this path, the $n = 20$ most recent note onsets which lie at least 1 second in the past are selected, and the local tempo for each onset is computed by considering the slope of the rectified path in a window with size 3 seconds centered on the onset. This results in a vector v_t of length n of relative tempo deviations from the score representation. Finally, an estimate of the current relative tempo t is computed using Eq.1, which emphasizes more recent tempo developments while not discarding older tempo information completely, for robustness considerations.

$$t = \frac{\sum_{i=1}^{n}(t_i * i)}{\sum_{i=1}^{n} i} \quad (1)$$

Of course, due to the simplicity of the procedure and especially the fact that only information older than 1 second is used, this tempo estimation can recognize tempo changes only with some delay. However, the computation is very fast, which is important for real-time applications, and it proved very useful for the task we have in mind.

4.2 Feeding Tempo Information to the ODTW

Based on the observation that both the alignment precision and the robustness directly depend on the similarity between the tempo of the performance and the score representation, we now use the current tempo estimate to alter the score representation on the fly, stretching or compressing it to match the tempo of the performance as closely as possible. This is done by altering the sequence of feature vectors representing the score audio. The relative tempo is directly used as the probability to compress or extend the sequence by either adding new vectors or removing vectors.

More precisely, after every incoming frame from the live performance, and before the actual path computation, the current relative tempo t is computed as given above, where $t = 1$ means that the live performance and the score representation currently are in the exact same tempo and $t > 1$ means that the performance is faster than the score representation. The current position in the score p_s is given by the forward path and thus coincides with the index of the last processed frame of the score representation. If a newly computed random number r between 0 and 1 is larger than t (or $\frac{1}{t}$ if $t > 1$) an alteration step takes place. If $t > 1$, a feature vector is removed from the score representation by replacing p_s+1 and p_s+2 with a mean vector of $p_s + 1$ and $p_s + 2$. And if $t < 1$, a new feature vector, computed as the mean of p_s and p_s+1 is inserted next into the sequence between p_s and p_s+1. As our system is based on features emphasizing note onsets, score feature vectors representing onsets (which are known from the score) are not duplicated, as more (and wrong) onsets would be introduced to the score representation. In such cases the alteration process is postponed until the next frame. Furthermore, to avoid that the system could get stuck at one frame, alterations may take place at most 3 times in a row.

5. 'LEARNING' TEMPO DEVIATIONS FROM DIFFERENT PERFORMERS

As will be shown later in Section 6, the introduction of this very simple tempo model – simply using the current estimated tempo to stretch/compress the reference score audio – already leads to considerably improved tracking results. But especially at phrase boundaries with huge changes in tempo (e.g. a slow-down or a speed-up by a factor of 2 is not uncommon, see also Figure 3) the above-mentioned delay in the recognition of tempo changes still results in large alignment errors. Furthermore, such tempo changes

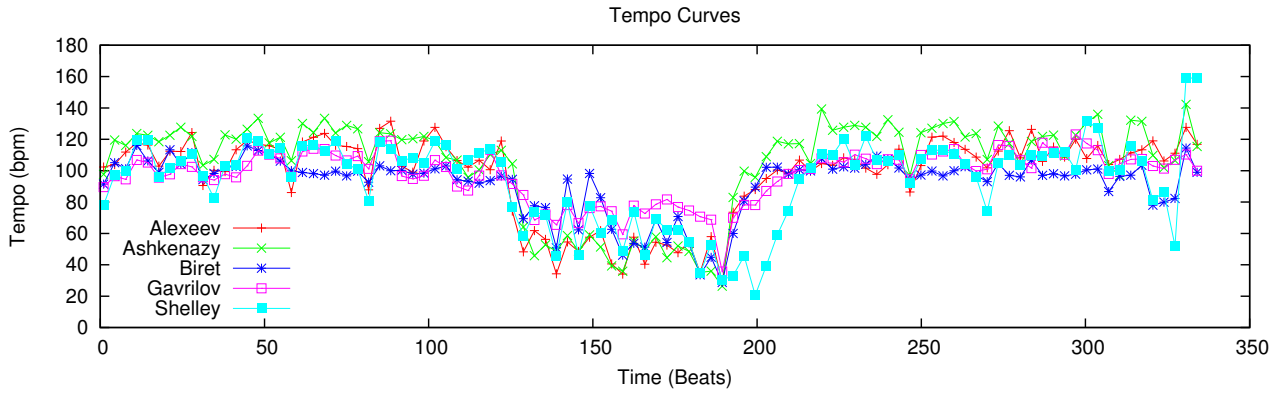


Figure 3. Tempo curves (at the level of quarter notes) automatically extracted from 5 different commercial recordings of the Prelude Op. 23 No. 5 by Rachmaninoff. Note especially the slow-down around beat 130 and the subsequent speed-up around beat 190 and the generally big differences in timing between the performances.

are very hard to catch instantly, even with more reactive tempo models. To cope with this problem we came up with an automatic and very general way to provide the system with information about possible ways in which a performer might shape the tempo of the piece.

First we extract tempo curves from various different performances (audio recordings) of the piece in question. Again, as for the real-time tempo estimation, this is done completely automatically using the method described in [6] (see Section 4.1), but as the whole performance is known beforehand and the tempo analysis can be done off-line there is now no need for further smoothing of the tempo computation. These tempo curves (see Figure 3) are directly imported into our real-time tracking system.

We use this additional information during the tracking process to compute a tempo estimate based not only on tracking information about the last couple of seconds, but also on similarities to other known performances. More precisely, as before, after every iteration of the path computation algorithm the vector v_t containing tempo information at note onsets is updated based on the backward path and the above-mentioned local tempo computation method. But now the tempo curve of the live performance over the last $w = 50$ onsets, again located at least 1 second in the past, is compared to the previously stored tempo curves at the same position. To do this all n tempo curves are first normalized to represent the same mean tempo over these w onsets as the live performance. The Euclidean distances between the curve of the live performance and the stored curves are computed. These distances are inverted and normalized to sum up to 1, thus now representing the similarity to the tempo curve of the live performance.

Based on the stored tempo curves our system can now estimate the tempo at the current position. As the current position should be somewhere between the last aligned onset o_j and the onset o_{j+1} to be aligned next, we compute the current tempo t according to Formula 2, where t_{i,o_j} and $t_{i,o_{j+1}}$ represent the (scaled) tempo information of curve i at onset o_j and o_{j+1} respectively, and s_i is the similarity value of tempo curve i.

$$t = \frac{\sum_{i=1}^{n}[(t_{i,o_j} + t_{i,o_{j+1}})s_i]}{2} \quad (2)$$

Intuitively, the tempo is estimated as the mean of the tempo estimates at these 2 onsets, which in turn are computed as a weighted sum of the (scaled) tempi in the stored performance curves, with each curve contributing according to its local similarity to the current performance. Please note that this approach somewhat differs from typical ways of training a score follower to follow a particular performance. We are not feeding the system with 'rehearsal data' by a particular musician, but with many different ways of how to perform the piece in question, as the analyzed performances may be by different performers and differ heavily in their interpretation style. The system then decides on-line at every iteration how to weigh the curves, effectively selecting a mixture of the curves which represents the current performance best.

6. EVALUATION

The precision of our system was thoroughly tested on various pieces of music (see Table 1), with very well known musicians like Vladimir Horowitz, Vladimir Ashkenazy and Daniel Barenboim amongst the performers. While we currently focus on classical piano music, to show the independence of specific instruments we also tested our system on an oboe sonata by Mozart and the 1^{st} movement of the 5^{th} symphony by Beethoven.

As for the evaluation reference alignments of the performances are needed, Table 1 also indicates how the ground truth data was prepared. For the performance excerpts of the Ballade Op. 38 No. 1 by Chopin (CB) we have access to very accurate data about every note onset ('matchfiles'), as these were recorded on a computer-monitored grand piano. For the performances of the 3 movements of Mozart's Sonata KV279 (MS) the evaluation is based on exact information about every beat time, which was manually compiled. The evaluation of the other pieces is based on off-line alignments produced by our system, which generally are much more precise than on-line alignments. We

ID	Composer	Piece Name	Instruments	# Perf.	Eval. Type
BF	Bach	Fugue BMV847	Piano	7	Offline Align.
BS	Beethoven	5^{th} Symphony, 1^{st} Movement	Orchestra	5	Offline Align.
CB	Chopin	Ballade Op. 38 No. 1 (excerpt)	Piano	22	Match
CW	Chopin	Waltz Op. 34 No. 1	Piano	8	Offline Align.
MO1	Mozart	Oboe Quartet KV370 Mov. 1	Oboe, Violin, Viola, Cello	5	Offline Align.
MO3	Mozart	Oboe Quartet KV370 Mov. 3	Oboe, Violin, Viola, Cello	5	Offline Align.
MS1	Mozart	Sonata KV279 Mov. 1	Piano	5	Beats
MS2	Mozart	Sonata KV279 Mov. 2	Piano	5	Beats
MS3	Mozart	Sonata KV279 Mov. 3	Piano	5	Beats
RP	Rachmaninoff	Prelude Op. 23 No. 5	Piano	5	Offline Align.
SI	Schubert	Impromptu D935 No. 2	Piano	12	Offline Align.

Table 1. The data set used for the evaluation of our real-time tracking system.

are well aware that this information is not guaranteed to be entirely accurate, but we manually checked the alignments for obvious errors and are quite confident that the results based on these alignments are reasonable, especially as evaluations of CB and MS based on these alignments led to very similar numbers compared to the evaluation on the correct reference alignments.

For all pieces we used audio files synthesized from publicly available 'flat' MIDI files with fixed tempo as score representation, only the MIDI representing the Beethoven Symphony contained sparse tempo annotations.

The evaluation took the form of a *cross-validation*. Every performance in our data set (Table 1) was aligned with 3 algorithms: the system introduced in [5] with only minor changes and optimizations; the system including the simple tempo model (Section 4); and the tempo model that has access to a set of possible performance strategies (Section 5). For the latter, all recordings pertaining to the given piece were used except, of course, for the performance currently being aligned. The result, for each performance and each algorithm, is a set of events with detection times in milliseconds.

The evaluation itself was performed as proposed in [7]. For each event i the difference (offset e_i) in milliseconds to the reference alignment is computed. An event i is reported as *missing* if it is aligned with $e_i > 250ms$. This percentage of notes thus misaligned (or, inversely, the percentage of correctly aligned notes) is the main performance measure for a real-time music tracking system. Further statistics, providing information about the alignment precision on those events that were correctly matched, and thus computed on e_i excluding missed events (ec_i), are the *average error*, defined as the mean over the absolute values of all ec_i, the *mean error*, defined as the regular mean without taking the absolute value, and the standard deviation of ec_i. Finally two measures are computed which sum up the overall performance of the system: the *piecewise precision rate* (PP) as the average of the percentage of correctly detected events for each group of performances (see Table 1) and the *overall precision rate* (OP) on the whole database.

Table 2 summarizes the results. Clearly, both tempo models lead to large improvements in tracking accuracy for pieces played with a lot of expressive freedom, especially for the Schubert Impromptu (SI), the Rachmaninov Prelude (RP) and the Chopin Waltz (CW), for which the number of missed notes is more than halved. Nonetheless these kinds of music still pose a great challenge to real-time tracking systems. As the results for the Beethoven Symphony (BS) show, our system can also cope quite well with orchestral music and does not depend on specific instruments. This is also supported by the results on the Oboe Quartet (MO).

As was to be expected, the results for pieces with less extreme tempo deviations were improved to a much smaller extent. Further investigation showed that as intended, the 'learned' tempo curves guided the alignment path more accurately and more reactively during huge tempo changes (i.e., at phrase boundaries).

Unfortunately it is not easy to make comparisons between different approaches in the literature, as the focus on a particular kind of music (e.g. contemporary vs. romantic piano music or monophonic vs. heavily polyphonic music) and the area of application (e.g. automatic accompaniment vs. visualization of music) have a huge influence on the design of the system. That makes it hard to compile a well-balanced ground truth database suitable for all systems.

With this in mind, and the fact that most of our results are currently only computed relative to off-line alignments as ground truth, we merely want to point out some observations. First there is an overlap between our data set and the one used for the evaluation of 'Antescofo' [2], which was already used professionally in a number of live performances. Using the same evaluation metrics, our system performed significantly better (1.9% vs. 9.33% missed notes) on the Fugue by Bach (BF). Of course the result for 'Antescofo' is based on only 1 single performance, which may not even be in our data set. Furthermore, we are quite sure that our system will perform significantly worse than 'Antescofo' on sparse monophonic data, as we do not explicitly detect note onsets and our forward path tends to 'randomly' wander around during long pauses between note onsets. Also, we allow our system to report notes early while 'Antescofo' is purely reactive, thus effectively giving our system twice as large a window to report onsets 'correctly'. While for the task of automatic accompani-

	No Tempo Model				Simple Tempo Model				'Learned' Tempo Model			
	Offset (ms)			%	Offset (ms)			%	Offset (ms)			%
ID	**Avg.**	**Mean**	**STD**	**Miss**	**Avg.**	**Mean**	**STD**	**Miss**	**Avg.**	**Mean**	**STD**	**Miss**
BF	52.1	-15.3	70.4	2.7%	41.7	0.1	61.3	2.2%	41.3	-0.3	59.3	1.9%
BS	84.1	4.3	106.5	15.9%	79.0	-11.6	100.8	15.0%	78.3	-6.4	100.3	13.9%
CB	63.1	16.6	83.7	10.9%	62.4	8.6	83.8	10.0%	63.1	3.9	85.2	9.9%
CW	86.3	-24.6	107.1	27.6%	78.7	-23.2	99.2	16.3%	75.4	-20.2	95.7	11.9%
MO1	94.8	-75.7	89.1	15.0%	70.1	-22.8	90.0	7.0%	72.1	-30.5	89.9	6.9%
MO3	99.9	-84.5	85.3	18.4%	64.3	-18.0	84.0	7.9%	65.7	-16.9	85.8	7.0%
MS1	47.4	13.8	64.5	3.6%	44.9	9.7	62.5	3.3%	42.7	10.1	59.5	3.2%
MS2	85.6	-21.3	104.8	19.8%	71.8	-4.7	93.7	13.8%	73.3	-6.4	94.5	11.3%
MS3	44.1	28.7	58.4	3.9%	40.2	6.7	59.5	3.3%	39.5	9.9	58.5	2.1%
RP	79.8	-18.7	102.0	31.8%	75.5	-10.5	96.8	17.1%	70.9	-10.6	93.2	14.8%
SI	107.3	-59.2	113.9	41.8%	77.9	-32.8	95.2	23.6%	78.7	-33.1	95.7	20.1%
OP	**83.2%**				**89.7%**				**91.1%**			
PP	**81.1%**				**87.9%**				**91.4%**			

Table 2. Real-time alignment results for all 3 evaluated systems (see text).

ment notes reported early are very bothersome, we think that for the task of real-time music visualization, which is our current focus, this is more tolerable.

Unfortunately, we could not find a comparable evaluation of 'Music Plus One' [1], which, like our system, focuses on classical music. However, a number of live demonstrations and available videos suggest that the system works very well in real-time accompaniment settings, not only reacting to tempo changes, but actually predicting them quite well.

That said, our real-time tracking system combines competitive alignment results with a unique feature not found in the above-mentioned systems: the ability to cope with arbitrary jumps of the performer(s) on-line by continuously tracking the performance at a coarser level and refining hypotheses about the current score position (see Section 3). This not only allows to, e.g., automatically cope with arbitrary rehearsal situations, where the musician(s) may keep repeating parts of the piece over and over, but effectively makes it impossible for the system to get lost. (Detailed experimental proof of that can be found in [3].)

7. CONCLUSION AND FUTURE WORK

We have presented a new approach to the incorporation of tempo information into a very robust real-time tracking system that is capable of dealing on-line with almost arbitrary structural deviations from the score. We demonstrated two ways to compute a tempo estimate, one only based on the alignment of the last couple of seconds of the performance, and one additionally based on a collection of previously extracted possible timing patterns, thus giving the system the means to anticipate tempo changes of the performer. The system was evaluated on a range of pieces from Western classical music. Both tempo models lead to significantly improved alignment results, especially for pieces played with a lot of expressive freedom.

An important direction for future work is the introduction of explicit event detection into our system, based on both an estimation of the timing and an analysis of the incoming audio frames. Furthermore we should think about ways to use the extracted tempo information to further improve the high level 'any-time' tracking process (not described in this paper – see [3]).

8. ACKNOWLEDGEMENTS

This research is supported by the City of Linz, the Federal State of Upper Austria, and the Austrian Federal Ministry for Transport, Innovation and Technology, and the Austrian Science Fund (FWF) under project number TRP 109-N23.

9. REFERENCES

[1] C. Raphael, "Current directions with music plus one," in *Proc. of the Sound and Music Computing Conference (SMC)*, (Porto, Portugal), 2009.

[2] A. Cont, "A coupled duration-focused architecture for realtime music to score alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, 2009.

[3] A. Arzt and G. Widmer, "Towards effective 'any-time' music tracking," in *Proc. of the Starting AI Researchers' Symposium (STAIRS 2010), European Conference on Artificial Intelligence (ECAI)*, (Lisbon, Portugal), 2010.

[4] S. Dixon, "An on-line time warping algorithm for tracking musical performances," in *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, (Edinburgh, Scotland), 2005.

[5] A. Arzt, G. Widmer, and S. Dixon, "Automatic page turning for musicians via real-time machine listening," in *Proc. of the 18th European Conference on Artificial Intelligence (ECAI)*, (Patras, Greece), 2008.

[6] M. Mueller, V. Konz, A. Scharfstein, S. Ewert, and M. Clausen, "Towards automated extraction of tempo parameters from expressive music recordings," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, (Kobe, Japan), 2009.

[7] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, "Evaluation of real-time audio-to-score alignment," in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR)*, (Vienna, Austria), 2007.

DANCE PATTERN RECOGNITION USING DYNAMIC TIME WARPING

Henning Pohl, Aristotelis Hadjakos
Telecooperation
Technische Universität Darmstadt
Darmstadt, Germany
{hpohl@rbg, telis@tk}.informatik.tu-darmstadt.de

ABSTRACT

In this paper we describe a method to detect patterns in dance movements. Such patterns can be used in the context of interactive dance systems to allow dancers to influence computational systems with their body movements. For the detection of motion patterns, dynamic time warping is used to compute the distance between two given movements. A custom threshold clustering algorithm is used for subsequent unsupervised classification of movements. For the evaluation of the presented method, a wearable sensor system was built. To quantify the accuracy of the classification, a custom label space mapping was designed to allow comparison of sequences with disparate label sets.

1. INTRODUCTION

Detecting patterns in movements is useful in a number of scenarios. Here, the focus is on dance movements in particular and their application in the area of interactive dance. In interactive dance human movements influence a computational system and motion patterns provide an additional capability in this process.

Interactive dance applications can mostly be found in the areas of art, gaming and clubs. For artistic purposes interactive dance is primarily used in installations, performance art and contemporary dance. Often dancers are given a certain level of control over the audio playing or the stage lighting. Dance is also used for generative art pieces to control a visualization. Video games, using interactive dance, mostly are interested in rhythmic patterns and how well a player adheres to a given step sequence.

The target scenario for this paper is the usage of interactive dance in a club setting. One open question in that context is how to enable more audience interaction and interactive dance is one possible way to do so. In this way it can provide an additional tool to DJs, VJs or other stakeholders in the overall experience. User studies with DJs and VJs have shown that they primarily assess a crowd's level of excitement and involvement using visual cues [1, 2]. Technological means to help with this assessment were generally viewed critically with participants fearing a loss of artistic freedom.

Recognizing dance patterns can serve two roles in this scenario. Such systems could, e.g., be used to gather information about dancers to provide an additional layer of information to DJs and VJs. A more interesting use however, is the possibility to outfit members of a DJ or VJ team with this technology. While being able to immerse themselves in the crowd they would still retain a level of control over the artistic process via their own movements. Pre-defined mappings or live mapping by a partner can be used to translate such movements to the desired auditive and visual output. Where other systems, e.g., provide continuous value activity measures, pattern information provides information in a discrete label space. Thus, pattern information can provide an additional informational layer to work with.

Interactive dance in a dance club context was explored by Ulyate and Bianciardi at the 1998 ACM SIGGRAPH convention, where they tested a number of prototypes [3]. They found that those devices that allowed for more freedom of movement, were yielding more satisfying dance interactions than devices such as buttons or pads. They also devised the role of an "experience jockey", who controls the overall experience and changes mappings according to the current situation.

Also interested in dance club interactions, Feldmeier did a user study on their viability and quality [4]. Several user tests on groups of up to 200 participants were done and showed an overall positive user response to the system. Participants enjoyed the experience itself and felt that the music and lighting adapted well to their motions.

2. RELATED WORK

This section discusses sensor options for capturing dance movements and algorithms that have used to analyze dance movement.

2.1 Sensors for Interactive Dance

There are a number of different approaches to make dance movements available to the computer.

2.1.1 Sensing floors

Sensing floors have been used to detect foot steps. Sensing floor systems vary with respect to resolution, modularity, size and response time. One of the first sensing floor system Johnstone's *PodoBoard* [5]. To use the system, metal plates had to be placed on the shoes of the users. Furthermore,

a pressure sensor was used to estimate the velocity of the shoe when touching the board. The *Magic Carpet* system was made of a grid of wires that were insulated with a piezoelectric material [6]. Furthermore, two doppler radars were present in the MagicCarpet system to detect upper body movement. The *LiteFoot* system detects lower body movements based on optical sensors, which are placed below the translucent floor [7]. The *Z-Tiles* is uses piezoelectric sensing to sense foot steps [8]. Similar to the LiteFoot, the Z-Tiles is assembled from modular plates that can be freely combined. Srinivasan et al. built a modular system based on pressure sensitive polymers [9].

2.1.2 Sensing Shoes

Paradiso & Hu built a sensing shoe that allowed to measure the pressure exerted by the toes and the heel with piezoelectric pads, the bending of the shoe with a force-sensitive strip, and the movements with inertial sensors [10]. The *Shadow Dancer* system was based on a stet dancing shoe equipped with pressure sensors at the tip and the heel [11]. Fujimoto et al. used a three-axis accelerometer to the tip of each shoe [12].

2.1.3 Camera-Based Systems

Bevilacqua et al. used a marker-based motion capture system to track the movements of dancers [13]. Castellano et al. analyzed the camera signals with the EyesWeb platform to determine the quantity of motion and the amount of space a dancer occupies [14]. Ng uses a camera to track dancers that wear color-coded costumes [15]. Guedes uses video data to detect the frequency spectrum of dancing movements [16]. For this purpose, the difference of luminance between two sequential frames is computed and input into an array of 150 band pass filters.

2.1.4 Wearable Sensors

Various sensing technologies have been used to build wearable sensors to record dance movements: Hromin et al. used accelerometers, flex sensors, temerature sensors, photoresistors and pressure sensors among others [17]. Aylward & Paradiso used inertial sensing [18]. El-Nasr & Vasilakos use a special armband that measures heat flux, skin temperature, near body temperature and galvanic skin response and heart rate [19]. Based on that data, the dancer's arousal state is estimated.

An important aspect of a wearable sensor system that is used to record dance movements is the communication protocol to transmit the data to the computer. Hromin et al. use Bluetooth [17] and Aylward & Paradiso developed their own wireless protocol [18]. The mentioned approaches have the advantage that they are power efficient, which helps to increase the uptime that can be achieved without changing batteries. The *WiSe Box* transmits the sensor data via WiFi [20]. The sensor values are packed in OSC messages before transmission.

2.2 Recognition Algorithms

Detecting patterns in motions or working with motions in general requires a way to detect similarities in motions. Algorithms doing so roughly fall into two categories:

Temporal Feature Classification Compare two motion sequences directly or using a set of descriptors

Non-Temporal Feature Classification Transform the data into a different space before further processing

With a way to determine similarities available, classification could be done via methods such as support vector machines (SVM), neural networks or the *k*-nearest neighbors (*k*-NN) algorithm.

2.3 Temporal Feature Classification

Gutknecht et al. used hidden markov machines (HMM) for Butoh, a form of experimental dance [21]. Setting out to classify movements, they designed a discreet three-dimensional motion space (*intensity*, *form* and *flow*) yielding a total of 64 motion categories. Dancers are equipped with three-axis accelerometers at the wrist, upper arm and upper leg, whose readings are relayed via Bluetooth. For classification, the values in a two second long sliding window are transformed into a sequence of features. On those features three HMMs (one for each dimension) are used to determine the most likely motion state sequence. The final motion state decision for the block is done using a majority vote. Gutknecht et al. furthermore designed a mapping from the motion space to a custom emotion space and subsequently derived a visualization from the detected emotional state.

Another method, somewhat similar to HMMs is dynamic time warping (DTW). The general idea is to compute the similarity of two given sequences that may differ in the temporal domain.

Tang et al. developed an algorithm to find repetitive patterns in motion capture data of dances [22]. 35 markers are tracked on a participant and the resulting posture data is normalized. From the resulting motion sequence a similarity matrix is derived. Postures in two frames are similar if the sum of the point by point euclidean distances is low. Repetitive motions can now be deducted from the similarity matrix, where diagonal patterns of similarity denote sequences, equally changing over time. Tracing patterns in the binarized similarity matrix in some respects is thus equal to similar image processing problems. Tang et al. use DTW to find such traces. Finally, using auto-clustering, patterns are classified as either cyclic or acyclic and an estimate of the cycle period is computed.

Fujimoto et al. used DTW to match foot movements to pre-recorded motion data [23]. Dancers perform their steps on top of a background track of constant rhythm. Based on the recognized dance steps, sounds are generated to go along with that background track.

DTW was also used by Bettens and Todoroff, who set out to detect gestures in a continuous sensor data stream [24]. For this purpose two sensors (three-axis accelerometer and two-axis gyroscope) are placed on both ankles of a dancing viola player. When performing, the downsampled sensor values are matched against a database of pre-recorded gestures. However, no segmentation of the live signal is tried.

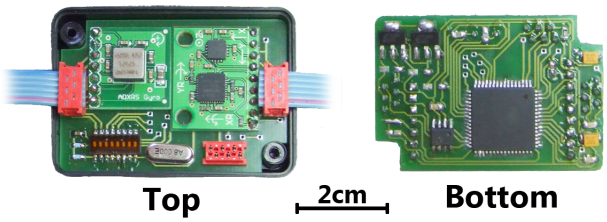


Figure 1. One of the used sensor nodes

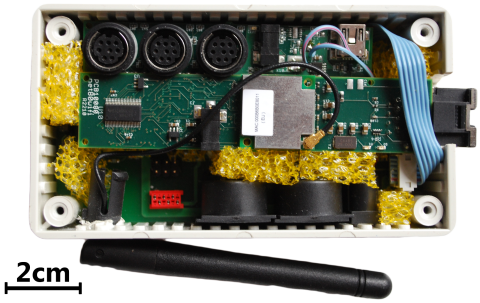


Figure 2. The interface board and the Gumstix

The signal is matched against the database at a number of different offsets instead.

2.4 Non-Temporal Feature Classification

Peng et al. built a system using two orthogonal cameras [25]. Based on training data, synthesized views are generated and used to build a tensor. During runtime the video data is similarly transformed into a corresponding tensor. Tensors are decomposed using higher order singular value decomposition, and used to approximate a view-invariant pose coefficient vector. With the coefficient vectors of the training set several classifiers were trained and subsequently evaluated. In their tests support vector machine (SVM) classifiers outranked fixed-threshold and von Mises-Fisher recognizers with recognition rates around 85% and a false detection rate of about 5%.

Nevada and Leman developed a method that is able to detect periodic movements in samba dance [26]. Two professional samba dancers were recorded on video, which was manually processed to derive a set of feature vectors. A periodicity transform [27] is used to find movements correlating with the musical meter. Thus, the proportion of periodicities in the signal is determined (independently for both dimensions). While no classification of gestures was performed, they noted that their approach could provide a useful methodology for dance analysis, with the periodicities potentially being used as classification features.

3. DANCESTIX

The *Dancestix* is a wearable sensor system that we developed and custom-built to record dance movements. The requirements for the Dancestix were:

- Measurement accuracy (R1),
- Usability in a dance club (R2), and
- Wearing comfort (R3).

The Dancestix consists of several inertial sensor nodes, a Gumstix embedded Linux system, and an interface board.

3.1 Inertial sensor nodes

The inertial nodes provide measurements of 3D linear acceleration and 3D angular velocity. The 3-axis accelerometer ADXL330 by Analog Devices was used to measure linear acceleration. Angular velocities were measured with the 2-axis gyroscope IDG-300 by InvenSense and the single-axis gyroscope ADXRS300 by Analog Devices. The measurements were performed at a resolution of 10 bit and a sampling rate of 100 Hz. The ADXL330 provides a measurement range of ± 3g, the IDG-300 provides $\pm 500°$/s and the ADXRS300 provides $\pm 300°$/s. The sensor nodes send the data to the interface board over CAN (Controller-area network). We built four of said sensor nodes. A sensor was worn around the hip, on the right upper arm, on the right forearm and on the right thigh. Further details on the inertial sensor nodes can be found in [28].

3.2 Interface board and Gumstix

The interface board connects the sensor nodes to the Gumstix embedded Linux system (a *Verdex Pro XM4* mainboard with *console-vx* and *netpro-vx* expansions). The interface board communicates with the sensors over CAN and transmits the sensor data to the Gumstix over a serial interface (RS232). Furthermore, the interface board acts as a central power supply for the inertial sensor nodes and the Gumstix. Wi-Fi is used for communication of the data from the Gumstix to a host computer.

3.3 Discussion

Because of the wired connection between the sensor nodes and the interface board, a local power supply at each sensor node is unnecessary. This helps to reduce the weight of the sensor nodes, which is beneficial for measurement accuracy (R1) as independent motion of the sensor because of its inertia is minimized. Furthermore, the reduced weight and size are beneficial for wearing comfort (R3). The disadvantage of wires running along the dancer's body could be eliminated by integrating the wires and sensors into a special clothing.

Bluetooth and ZigBee were among the options that were considered for wireless transmission of the sensor data to a host computer. Bluetooth and ZigBee have the advantage that they consume significantly less power than Wi-Fi, which would maximize battery lifetime. However, Bluetooth and ZigBee are intended for wireless personal area networks (WPAN) with operating spaces of typically 10 m [29], which would be too little to be usable in a dance club (R2). Furthermore, the number of sensor-equipped dancers would be too limited if Bluetooth or ZigBee were

used. Bluetooth allows only seven slaves per master and ZigBee has a communication bandwidth of only 250 kb/s while a single Dancestix produces 24kb/s of data (4 sensors, 6 DoF/sensor, 100 samples/s, 10 bit/sample).

4. DANCE PATTERN RECOGNITION

When searching for patterns we are doing so on the measure level. While not all patterns in all forms of dance conform to this division, it is an appropriate choice for the target scenario. According to this choice, the dance data is to be split into blocks, each corresponding to one measure.

Segmenting motion data into measures is not an easy task though. While some approaches exist to perform beat detection on motion data, e.g., the work by Enke [30], detecting beats in audio tracks is far more accurate. We used *BeatRoot* by Dixon [31] to compute the beat timestamps for the audio track beforehand. As the audio is in $^4/_4$ format, four beats are automatically grouped to form one measure. The resulting blocks of motion data are multi-dimensional sequences of samples. The set of all blocks is denoted as $\mathbb{B}$.

4.1 Block Similarities

Finding reoccurring patterns ultimately is a problem of finding similar blocks. If the difference between two blocks is sufficiently small, chances are the movement in the second one is a reiteration of the first one. There are two problems at hand: how to determine the similarity of two blocks and what threshold to use when grouping them together. While the second problem requires some evaluation and is detailed later on, the first shall be described in this section.

We determine block similarities via the DTW algorithm [32]. In DTW, a mapping from an input sequence to a given sequence is found that minimizes the distance between them (using the euclidian distance metric for sequence element distances). The sequences do not need to be of equal length, as DTW corrects for differences in speed, but need to be of the same dimensionality. Running the DTW algorithm on two sequences yields two measures: a distance between those two sequences, and a so-called *warp path* that describes the best possible alignment of the two sequences. The best possible alignment is that alignment, which minimizes the overall distance between the two sequences.

To speed up the DTW computations one can use constraints to limit the amount of calculations needed. We used *FastDTW* by Salvador and Chan [33] instead of a basic DTW. This algorithm reduces DTW complexity by iteratively computing the DTW for a coarser resolution, projecting it to a finer one and refining it. Using that multilevel approach, *FastDTW* works in $\mathrm{O}(n)$, similar to constrained versions of DTW. Due to the nature of the algorithm a certain level of error is induced. This is primarily dependent on the radius used. Analysis by Salvador and Chan showed, that for higher radii the error converges to 0. Furthermore, this convergence is significantly faster than in constraint based DTW implementations. An analysis of appropriate *FastDTW* radii for the movement data used here, is given in Section 6.

4.2 Classification

With a way to compute block similarities, further processing steps have to be taken to classify blocks into distinct groups. As no *a priori* knowledge on possible patterns is assumed, for maximal flexibility during live performances, classification is unsupervised and solely depends on statistical information. Furthermore, classification should work in real-time on streams of incoming motion data blocks. Unfortunately, having no set of labels given and the requirement to assign new labels to blocks as they come in, imposes severe restrictions on the classification. For example, a simple *k*-NN clustering would not work, as the number of desired clusters *k* is not known. Changing assigned labels later on is also undesirable, as such a change could confuse label consumers. For example, VJs using class information should be able to rely on consistent labels when using them to enrich their work. Thus, algorithms determining clusters by reexamining the classes for all available data from the set do not work. An algorithm is needed that preserves previous class assignments and classifies new blocks based on previous classification choices.

We used a threshold clustering approach for this purpose that assigns class labels on-the-fly and unsupervised. Assigned class labels are immutable as required above. In this approach, an input stream of motion data blocks shall be modeled as a sequence S, with

$$S = (s_1, s_2, s_3, \ldots) \quad \text{with } s_n \in \mathbb{B}. \tag{1}$$

Clusters of blocks are modeled as sets C and represent blocks that were grouped together. All clusters are stored in sequence L, where a cluster's index in this sequence also serves as class label.

For the first incoming block $S(1)$, there is no decision to be made. It is the foundation for a first cluster, thus resulting in

$$L = (\{S(1)\}). \tag{2}$$

For subsequent blocks, the best matching preexisting cluster has to be determined. This assumes a distance function $dist$ is defined, which for the purposes of this paper will be the DTW algorithm. In addition to that function, a threshold t has to be provided as well. The distance from a new block $S(i)$ to an already found cluster C in L, is defined as the average distance to C's members.

$$clusterDistance\,(S(i), C) = \frac{1}{|C|} \sum_{c \in C} dist\,(S(i), c) \tag{3}$$

The best match of a new block is hence given as the cluster with the smallest distance to, from all available clusters.

$$\begin{aligned} &bestMatch\,(S(i), L) = \arg\min\{ \\ &\quad clusterDistance\,(S(i), L_n) : 1 \leq n \leq |L| \\ &\} \end{aligned} \tag{4}$$

Based on the given threshold t, new blocks are either assigned to the best matching cluster or not. If the distance from the block to its best matching cluster is lower than t it is considered to belong to it. If that is not the case, a new cluster is created based on that block.

The threshold has to be chosen carefully for good results. If it is set too low, all blocks are considered unrelated and are assigned their own class. Conversely, a threshold value that is too high, leads to undesired mashing of blocks that should have been distinguished. The threshold value is dependent on the moves used and the features chosen. In Section 6, results for different threshold values are shown in detail.

5. EVALUATION METHOD

The proposed algorithm was evaluated in a user study with four participants. A remix of Lady Gaga's song "*Just Dance*" was used in this study. At 119 bpm it is moderately fast and fits into the desired target scenario. The length of 4:54 min (which corresponds to 145 measures) is long enough to allow for several different dance patterns to be tested in a realistic setting. With a prominent bass drum track it also allows for accurate beat detection and makes it easy for participants to stay on time.

Six different dance movements were defined and assembled into a choreography for the study:

A. Side steps with no arm movement

B. Rock steps sideways without arm movement

C. Rock steps sideways with arm movement

D. Side steps with arm movement

E. Side steps with arms up in the air

F. Standing still with head bopping

There are two distinct foot movements combined with three possible arm movements. In addition, there is a resting pose, used in three short intermissions present in the song. The movements were chosen as to allow participants to take part in the evaluation without lengthy training and for their good fit to a club setting. More complex movements were deemed too difficult for non-professional dancers in an ad hoc evaluation session.

Before starting a recording session, all participants were instructed on the testing procedure. The dance moves to be performed were explained to them beforehand as well. While recording, additional help was provided in the form of oral notification of upcoming transitions and movement instructions.

5.1 Classification Quality Rating

Running the proposed algorithm on the recorded dance data yields a sequence of class labels. The choreography given to the participants also defines a sequence of classes. When rating the classification quality the matching between those two sequences has to be determined. As the classifier labels blocks without a priori label information the set of labels used in both sequences will be disparate. Hence, a best fit label space mapping algorithm was used to align the label sequences.

Given are two label sequences A and B with corresponding label spaces $\mathcal{A}$ and $\mathcal{B}$:

$$\begin{aligned} A &= (a_1, a_2, a_3, \ldots) \quad a_n \in \mathcal{A} \\ B &= (b_1, b_2, b_3, \ldots) \quad b_n \in \mathcal{B} \end{aligned} \tag{5}$$

To compare A and B we need a function that maps $\mathcal{A}$ to $\mathcal{B}$. This function should be injective as to not allow multi mappings to the same label:

$$\begin{aligned} f &: \mathcal{A} \to \mathcal{B} \\ f(a_1) &= f(a_2) \Rightarrow a_1 = a_2 \end{aligned} \tag{6}$$

To find this mapping a cost matrix $\mathbf{C}$ is used. This matrix of size $|\mathcal{A}| \times |\mathcal{B}|$ is initialized to the zero matrix. Now both label sequences are traversed and the cost matrix is updated accordingly:

$$\begin{aligned} \mathbf{C}^i &= \mathbf{C}^{i-1} + \mathbf{D}^i \\ \mathbf{D}^i_{x,y} &= \begin{cases} 1 & \text{if } a_i = \mathcal{A}_x \wedge b_i = \mathcal{B}_y \\ 0 & \text{if } a_i \neq \mathcal{A}_x \\ -1 & \text{if } a_i = \mathcal{A}_x \wedge b_i \neq \mathcal{B}_y \end{cases} \end{aligned} \tag{7}$$

In the final cost matrix higher values indicate good fits. The mapping can then be extracted by reducing the matrix. Before doing so, however, the matrix is normalized with the corresponding class frequencies.

In each reduction step that mapping is found in the matrix, which has the best fit. This is denoted by the highest cell value in the cost matrix. The row and column of said mapping are then eliminated from the matrix. This process continues while there are still open classes to be mapped from and mapped to.

After this reduction a mapping from one label set to the other is found. For sets differing in size, some classes will not map to another one or will not be mapped to. The quality of the results from a classification can thus now be quantified via simple equality testing.

6. RESULTS

Based on the recorded motion data from several participants, the performance of the DTW algorithm was evaluated. Several aspects are of interest at this point:

- How well can distinct movements be distinguished?
- What is the influence of parameter choices?
- How strong is the influence of the sequence length?

The error rate is sometimes deceiving. Consider an algorithm that assigns a new label to each block. As the identifier mapping (see Section 5.1) tries to find the best fit, some blocks are still seen as correct. Specifically, an amount of blocks equal to the number of distinct classes in the given sequence is considered correct. Applying that logic to a sequence of 20 blocks with four distinct given classes, 20% would be seen as correctly classified even though no blocks were grouped. For longer sequences, this problem becomes less of an issue, as long as the number of distinct given classes does not increase correspondingly.

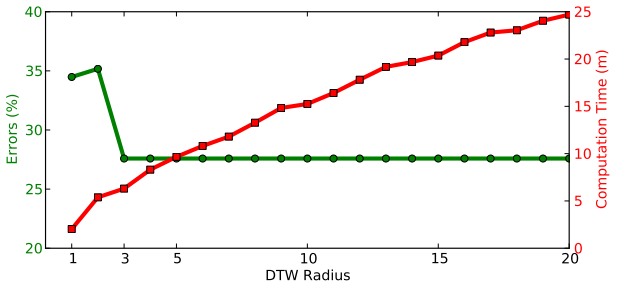


Figure 3. Comparing cost and error rate of different *FastDTW* radius choices.

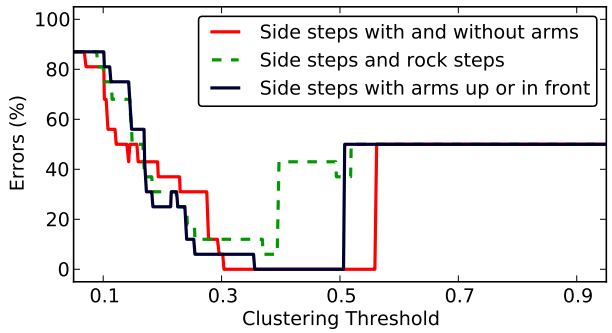


Figure 4. Comparison of error rate when detecting differences between pairs of movement.

The other extreme is a classification that assigns all blocks to the same class. The identifier mapping will connect that class with the given class of the highest frequency. To put this in perspective, consider a 20 block sequence with one 10 block spanning class and two classes spanning 5 blocks. A recognized sequence of only one class would map to the 10 block long given sequence. Thus, 50% of the data is considered as correctly classified. This problem becomes less of an issue when the number of distinct given classes goes up or given classes are more uniformly distributed.

6.1 Dynamic Time Warping Radius

In a first step, appropriate radii for use with DTW were determined. As mentioned in Section 4.1, this value influences the computational cost and accuracy of the DTW algorithm. A multitude of DTW radii were tested on a full dance recording (145 blocks), with results shown in Figure 3. As can be seen any radius higher than 3 does not result in less errors. The computational cost increases significantly, though. Based on this result, a DTW radius of 4 was chosen for all subsequent DTW calculations.

6.2 Pairwise Motion Comparison

To determine the suitability of DTW, it was tested on a sequence of two different motion patterns. Sixteen blocks from a recorded session were used, spanning about 32 seconds in time. Each motion in the pair to be tested spans half of those blocks. As can be seen in Figure 4, our method was

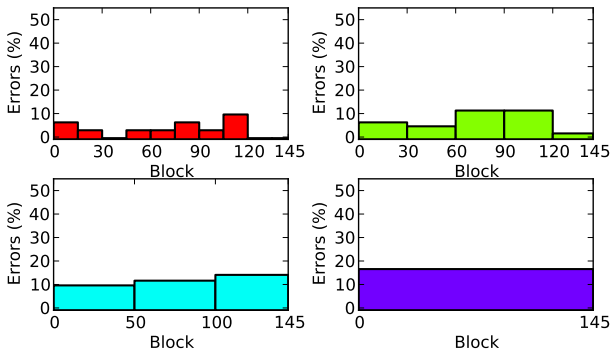


Figure 5. Classification results for several subsequences of sensor data. The width of each bar denotes the data range being used.

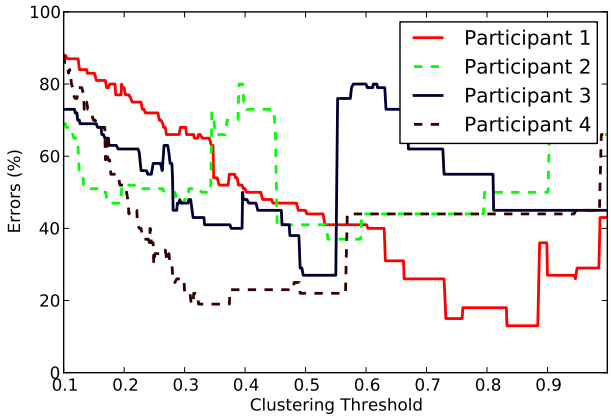


Figure 6. Comparing classification results from multiple participants.

able to correctly differentiate between motion pairs. Note that there is a range of threshold values being appropriate and some pairs are easier to differentiate than others.

6.3 Influence of Sequence Length

The length of a sequence has an influence on the classification accuracy. One aspect is that over time, slight changes to a dance move are more likely to occur. While it is comparatively easy to perform the same move for 20 seconds, it is harder to repeat the same move after 2 minutes have passed. Thus, one would expect that a classification of shorter time spans contains less errors. To test this, one recording was analyzed in various windows. Looking at the results in Figure 5, a certain increase in overall error is apparent. While analyzing sequences of 15 block length, the maximum error was below 20%, the overall error in a classification of the whole sequence was at slightly over 30%. The data also shows peaks in error rate in later parts of the recorded sequence. Some of that can be attributed to the sensor straps loosening and required fastening motions.

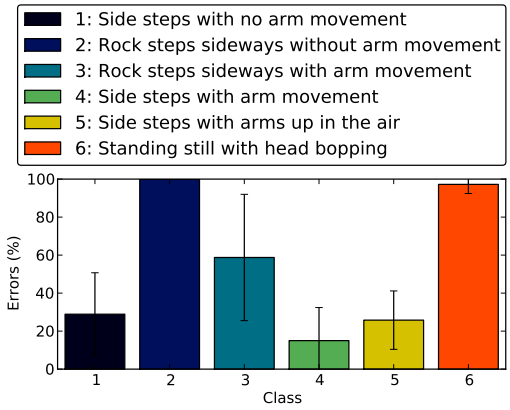


Figure 7. Comparing multi-participant error rates by given class

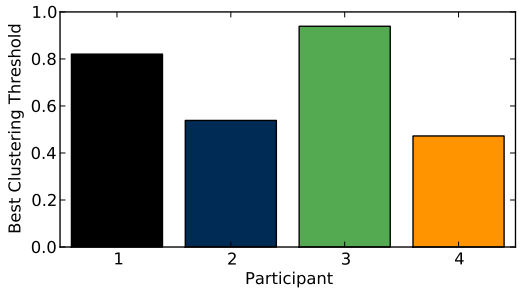


Figure 8. Means of lowest error yielding thresholds per participant

6.4 Comparison of Several Recordings

Figures 6 shows how well the DTW approach was able to classify recordings from four participants. As can be seen, some participant's motions were more easily discernable than other's. Also, each participants seems to have a separate best performing threshold. However, all curves exhibit somewhat similar behavior over the threshold range.

6.5 Comparison of Error Rate by Dance Move

While an overall error measure provides a general performance estimate, determining the error per dance move helps with a more in-depth understanding. Figure 7 shows such data for all participants. The variance in error rate is deceiving to some extent, as the different recordings have different base error rates. Thus, the relative error rates are the most interesting aspect here. Looking at the data, classes 2 and 6 seem to be an issue. Those two coincidentally are also the two classes with the least amount of motion being required for them. On the other hand, more vivid movements were detected comparatively well.

6.6 Threshold Choice

In most previous comparisons the error rate was given as a function of the threshold being used. It could also be seen in Section 6.4 how the chosen threshold varies with each participant and performs best at different ranges. In Figure 8, a comparison of the best performing thresholds is shown. As can be seen, the range of appropriate thresholds is roughly contained in the $[0.5, 0.8]$ interval.

7. CONCLUSION

As shown in Section 6, DTW based classification is able to correctly distinguish two given motions in one sequence. Working with data from real dance recordings, error rates of about 20-30% have been achieved. The results indicate that some movements were harder to distinguish than others. Especially movements eliciting low sensor responses were problematic. However, more pronounced movements were recognized much better.

8. REFERENCES

[1] C. Gates, S. Subramanian, and C. Gutwin, "DJs' perspectives on interaction and awareness in nightclubs," in *Proceedings of the 6th ACM conference on Designing Interactive systems - DIS '06*, (New York, NY), ACM Press, 2006.

[2] A. Engström, M. Esbjörnsson, and O. Juhlin, "Mobile collaborative live video mixing," in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services - MobileHCI '08*, (New York, NY), ACM Press, 2008.

[3] R. Ulyate and D. Bianciardi, "The Interactive Dance Club: Avoiding Chaos in a Multi-Participant Environment," *Computer Music Journal*, vol. 26, no. 3, 2002.

[4] M. C. Feldmeier, *Large group musical interaction using disposable wireless motion sensors*. Master thesis, Massachusetts Institute of Technology, 2002.

[5] E. Johnstone, "A MIDI Foot Controller - The PodoBoard," in *International Computer Music Conference*, (San Francisco, CA), 1991.

[6] J. A. Paradiso, C. Abler, K.-y. Hsiao, and M. Reynolds, "The Magic Carpet: Physical Sensing for Immersive Environments," in *Conference on Human Factors in Computing Systems*, (New York, NY), ACM Press, 1997.

[7] N. Griffith and M. Fernström, "LiteFoot: A floor space for recording dance and controlling media," in *Proceedings of the 1998 International Computer Music Conference*, (Ann Arbor, MI), 1998.

[8] L. McElligott, M. Dillon, K. Leydon, B. Richardson, M. Fernström, and J. A. Paradiso, "'ForSe FIElds' - Force Sensors for Interactive Environments," in *UbiComp 2002: Ubiquitous Computing* (G. Borriello and L. E. Holmquist, eds.), vol. 2498 of *Lecture Notes in Computer Science*, (Göteborg, Sweden), Springer, 2002.

[9] P. Srinivasan, D. Birchfield, G. Qian, and A. Kidané, "A pressure sensing floor for interactive media applications," in *ACE '05: Proceedings of the 2005 ACM*

SIGCHI International Conference on Advances in computer entertainment technology, vol. 265, (Valencia, Spain), ACM, 2005.

[10] J. A. Paradiso, K.-y. Hsiao, and E. Hu, "Interactive music for instrumented dancing shoes," in *Proceedings of the 1999 International Computer Music Conference*, (Beijing, China), 1999.

[11] Y. Kim, D. Jung, S. Park, J. Chi, T. Kim, and S. Lee, "The Shadow Dancer: A New Dance Interface with Interactive Shoes," in *International Conference on Cyberworlds*, (Hangzhou, China), IEEE Computer Society, 2008.

[12] M. Fujimoto, N. Fujita, Y. Takegawa, T. Terada, and M. Tsukamoto, "Musical B-boying: A Wearable Musical Instrument by Dancing," *Lecture Notes In Computer Science*, vol. 5309, 2008.

[13] F. Bevilacqua, L. Naugle, and I. Valverde, "Virtual dance and music environment using motion capture," in *IEEE Multimedia Technology and Applications Conference Proceedings*, (Irvine, CA), 2001.

[14] G. Castellano, R. Bresin, A. Camurri, and G. Volpe, "Expressive Control of Music and Visual Media by Full-Body Movement," in *Proceedings of 2007 Conference on New Interfaces for Musical Expression*, (New York, NY), 2007.

[15] K. C. Ng, "Music via Motion: Transdomain Mapping of Motion and Sound for Interactive Performances," *Proceedings of the IEEE*, vol. 92, 2004.

[16] C. Guedes, "Extracting Musically-Relevant Rhythmic Information from Dance Movement by Applying Pitch Tracking Techniques to a Video Signal," in *Proceedings of the 2006 Sound and Music Computing Conference*, (Marseille, France), 2006.

[17] D. Hromin, M. Chladil, N. Vanatta, D. Naumann, S. Wetzel, F. Anjum, and R. Jain, "CodeBLUE: a bluetooth interactive dance club system," in *GLOBECOM '03. IEEE Global Telecommunications Conference*, vol. 5, (San Francisco, CA), IEEE, 2003.

[18] R. Aylward and J. A. Paradiso, "A compact, high-speed, wearable sensor network for biomotion capture and interactive media," in *Proceedings of the 6th international conference on Information processing in sensor networks - IPSN '07*, (Cambridge, MA), ACM, 2007.

[19] M. S. El-Nasr and T. Vasilakos, "DigitalBeing — Using the Environment as an Expressive Medium for Dance," *Information Sciences*, vol. 178, 2008.

[20] E. Fléty, "The Wise Box: a multi-performer wireless sensor interface using WiFi and OSC," in *Proceedings of the 2005 conference on New interfaces for musical expression*, National University of Singapore, 2005.

[21] J. Gutknecht, I. Kulka, P. Lukowicz, and T. Strieker, *Advances in Expressive Animation in the Interactive Performance of a Butoh Dance*, pp. 418–433. Springer, 2008.

[22] K.-T. Tang, H. Leung, T. Komura, and H. P. H. Shum, "Finding repetitive patterns in 3D human motion captured data," *Conference On Ubiquitous Information Management And Communication*, 2008.

[23] M. Fujimoto, N. Fujita, Y. Takegawa, T. Terada, and M. Tsukamoto, "A Motion Recognition Method for a Wearable Dancing Musical Instrument," in *2009 International Symposium on Wearable Computers*, IEEE, 2009.

[24] F. Bettens and T. Todoroff, "Real-time dtw-based gesture recognition external object for max/msp and puredata," in *Proceedings of the 6th Sound and Music Computing Conference*, (Porto, Portugal), 2009.

[25] B. Peng, G. Qian, and Y. Ma, "Recognizing body poses using multilinear analysis and semi-supervised learning," *Pattern Recognition Letters*, vol. 30, no. 14, 2009.

[26] L. Naveda and M. Leman, "Representation of Samba dance gestures, using a multi-modal analysis approach," in *5th International Conference on Enactive Interfaces*, (Pisa, Italy), Edizione ETS, 2008.

[27] W. A. Sethares and T. W. Staley, "Periodicity transforms," *IEEE transactions on Signal Processing*, vol. 47, no. 11, 1999.

[28] A. Hadjakos, E. Aitenbichler, and M. Mühlhäuser, "SYSSOMO: A Pedagogical Tool for Analyzing Movement Variants Between Different Pianists," in *5th International Conference on Enactive Interfaces*, (Pisa, Italy), Edizione ETS, 2008.

[29] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007.

[30] U. Enke, *DanSense: Rhythmic Analysis of Dance Movements Using Acceleration-Onset Times*. Master thesis, RWTH Aachen University, 2006.

[31] S. Dixon, "Evaluation of the Audio Beat Tracking System BeatRoot," *Journal of New Music Research*, vol. 36, 2007.

[32] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, 1978.

[33] S. Salvador and P. Chan, "FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space," in *KDD Workshop on Mining Temporal and Sequential Data*, (Seattle, WA), 2004.

Analysis and automatic annotation of singer's postures during concert and rehearsal

Maarten Grachten **Michiel Demey** **Dirk Moelants** **Marc Leman**
Institute of Psychoacoustics and Electronic Music (IPEM)
Department of Musicology – Ghent University
Ghent, Belgium
http://www.ipem.ugent.be

ABSTRACT

Bodily movement of music performers is widely acknowledged to be a means of communication with the audience. For singers, where the necessity of movement for sound production is limited, postures, i.e. static positions of the body, may be relevant in addition to actual movements. In this study, we present the results of an analysis of a singer's postures, focusing on differences in postures between a dress rehearsal without audience and a concert with audience. We provide an analysis based on manual annotation of postures and propose and evaluate methods for automatic annotation of postures based on motion sensing data, showing that automatic annotation is a viable alternative to manual annotation. Results furthermore suggest that the presence of an audience leads the singer to use more 'open' postures, and differentiate more between different postures. Also, speed differences of transitions from one posture to another are more pronounced in concert than during rehearsal.

1. INTRODUCTION AND RELATED WORK

The performance of music is naturally accompanied by corporal gestures of the performing musician. The form and roles of these gestures in music performance appear to be heterogeneous and have led to considerable investigation [1, 2, 3]. A distinction has been made between four categories of gestures: (a) sound-producing gestures, necessary to create sound; (b) communicative gestures between musicians or between the musician and the audience; (c) sound-facilitating gestures, which accompany the first category; and (d) sound accompanying gestures like dancing, that are generally not produced by the musician himself [4, 5]. In singing performance sound-producing gestures are very limited, the actual sound is produced inside the body and the only visually perceivable elements are the articulation of the mouth and possibly the breathing. Sound-facilitating gestures are equally limited and restricted to posture changes that facilitate singing in e.g. high or low registers. Most of the gestures perceived in singing performance can therefore be attributed to communication, and more specifically to communication with the audience. It can be shown through subtle gestures like facial expressions, but is most obvious in the movements of hand and arms and to a lesser extend the head and the upper body. These gestures may for example reflect the temporal structure of the piece, convey the mood of the piece, or underline the meaning or importance of certain words or phrases. Therefore the study of gestural aspects of singing performance can give us information on structural elements of the music, as well as on expression and communication.

Many studies on perceived expressivity in musician's gestures exist, based on direct measurements as well as on observations (for an overview see [5]), however these studies almost entirely deal with instrumentalists. Despite the importance of gestural communication in singing performance, studies on this topic are scarce. If we want to study gestural communication between a singer and the public, we can compare a performance in a dress rehearsal and in the actual concert. The performance during the dress rehearsal is supposed to be technically and interpretatively mature and will be performed in the same setting and the same order as the concert. The one aspect that is clearly different is the presence of a public. In general performing musicians acknowledge that the interaction with the public affects their performance, but very little is known about what is actually changing and how. Intuitively one could say that musical elements like the timing and dynamics change, but also the gestural communication is changing, using different movements, facial expressions or eye contact. By using multi-modal measurements (audio, video, movement sensors) and the development of new analytical techniques, we can quantify different aspects of the performance and thus develop a set of parameters that can be used to compare performances, in casu to detect the differences between a rehearsal and a concert performance.

This paper focuses on a singer's postures and the transitions between them. Techniques that allow automatic classification of a set of typical postures are presented and applied to the comparison of the recordings from the dress rehearsal and the concert.

2. DATA

A dress rehearsal and a concert by singer Chia-Fen Wu and viola da gamba player Dirk Moelants were recorded. The

Index	Composer	Piece
01	Giulio Caccini	Dolcissimo Sospiri
02	Giulio Caccini	Movetevi a pieta
03	Barbara Strozzi	Moralit amorosa
04	Barbara Strozzi	Non occore
05	Richard Sumarte	Daphne
06	John Dowland	Come Again
07	John Dowland	Flow my tears
08	Robert Johnson	Hark, hark, the lark
09	Tobias Hume	Tobacco
10	Thomas Morley	It was a lover and his lass
11	Richard Sumarte	What if a day
12	Richard Sumarte	Whoope doe me no harme
13	Henry Purcell	How sweet it is to love
14	Henry Purcell	Music for a while
15	Henry Purcell	If music be the food of love
16	Teng Yu-Hsien	Bang Chun Hong
17	Yang San-Lang	Go Luan Hue
18	traditional Chinese	Ye Lai Shiang

Table 1. Overview of the concert program analyzed in this paper. The pieces will henceforth be referred to by the numbers on the left

Figure 1. Typical examples of the 10 different categories of postures as found in the singers performance

program is given in table 1. Three pieces (05, 11 & 12) are short pieces for solo viola da gamba, they will not be considered in the present study. All the other pieces are performed by a (soprano) voice with viola da gamba accompaniment. The first 15 pieces are period-style arrangements of 17th century baroque music. The last piece (18) is a traditional Chinese song, in the concert performance it was brought as an encore. The two pieces before (16 & 17) are Taiwanese art songs from the middle of the previous century.

Three different measurements were made of the two performances: an audio recording, a measurement of the movement and a video recording. The audio was recorded using a mobile recorder with a built-in microphone (Zoom H2) positioned at the side of the stage. The movement of the performers was measured using wireless accelerometers with a range of +/-3g and with 2 or 3 sensitive axes. The singer had a sensor on each wrist and one sensor on her back. The sensors were attached to the skin with medical bandage tape underneath the clothes in such a way that they did not hamper the movements of the performers and that they were not visible for the audience. The accelerometers were connected to a standalone, battery powered, wireless ADC module (Wi-microDig, Infusion Systems) that digitizes the analogue sensor data and transmits this data wireless via Bluetooth. A Bluetooth class 1 interface was used enabling a range of 100m making it possible to collect the data from the balcony in the back of the concert hall. The sensor data was recorded at a sampling rate of 100Hz using a Max/MSP patch. Furthermore, the entire concert and rehearsal was videotaped using a Canon HV30 camera.

3. DATA ANALYSIS

3.1 Posture occurrences and durations

A first step in the analysis was the creation of a ground-truth of the postures used by the singer. First all the video material was watched to determine the different postures used. In total 10 categories of postures could be distinguished: 0. Arms down, hands joined, 1. hands joined in front of the body, 2. both arms slightly spread in front of the body, 3. both arms in front of the body, left arm above right arm, 4. both arms in front of the body, right arm above left arm, 5. right arm next to the body, left arm in front, 6. left arm next to the body, right arm in front, 7. two arms next to the body, 8. one arm next to the body, the other spread open and 9. two arms spread out. Examples of each posture, taken from the video recording are shown in figure 1.

In a next stage these 10 postures were used in a detailed manual analysis done by author DM using the program Annotation (`http://www.saysosoft.com`). A global overview of the postures used in the rehearsal and the concert is given in table 2. This shows that the singer changed posture more often during the concert, reflected in a increase of total postures of 9.75%. Postures 0, 8 and 9 are not very common, but still it is striking that they only occur during the concert performance. In figure 2 the representation is a bit simplified by grouping these three gestures as 'others' and by adding the symmetrical postures 3/4 and 5/6 together. The upper two pie diagrams represent the number of postures counted (analogous to table 2), the lower two represent the total time of each posture type. It shows that the most important change occurs in the number of and time spent in posture 1, that is hands joined in front of the body. This posture is considered as a 'resting' or 'starting' posture, and can thus be seen as the least 'communicative' or 'expressive' posture. This posture is largely replaced by more 'open' postures in the concert performance (types 5, 6 and 7).

As we count more stable postures in the concert performance, this implies that there are more transitions. In total the number of transitions increases with 10.26%, from 341 to 376. Despite the larger number of transitions, we see that the average duration of a transition increases with

posture	rehearsal	concert
0	0	3
1	57	27
2	65	71
3	37	31
4	3	1
5	36	86
6	43	26
7	118	141
8	0	1
9	0	7
sum	359	394

Table 2. Comparison of the number of posture occurrences for each of the 10 categories in rehearsal and concert performance, summed over all pieces

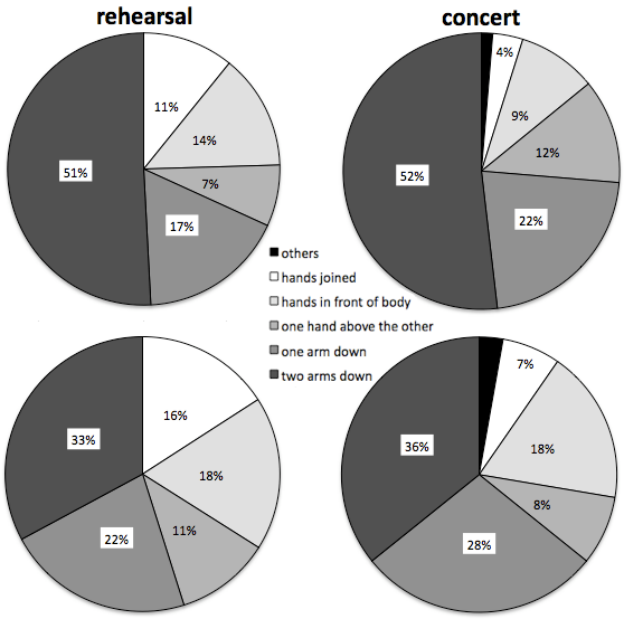


Figure 2. Top row: The relative occurrence of each posture (number of instances); Bottom row: Cumulative duration of each posture

6.45%, from 1.61s to 1.72s. However, as the distribution of the posture types in both performances is different (cf. supra), it is dangerous to make such a general comparison. Therefore the 20 most important transitions, occurring at least twice in each performance where selected. The average length of these transitions in both performances is shown in figure 3. It shows that there is often a large increase in average duration, while only four transition types show a small decrease in average duration. The grand average of these 20 transitions increases with 21.62% from 1.54s to 1.87s between rehearsal and concert performance. This shows that the singer puts more emphasis on the transitions, by making them slower or by increasing the distance.

3.2 Posture and transition analysis

For a further quantification of the postures by the singer the 3D accelerometer sensor data is analyzed. Starting from the 3D acceleration measured at the wrists of the singer it

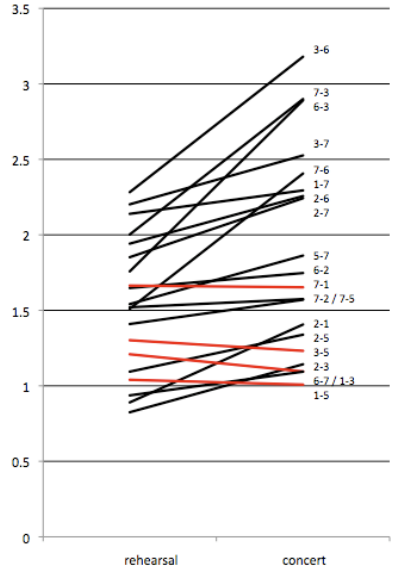


Figure 3. Differences in transition durations between rehearsal and concert, for the 20 most frequent transitions between postures

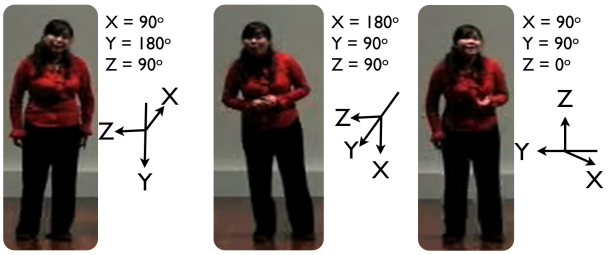


Figure 4. Three postures with the orientation of the 3D accelerometer on the left arm

is possible to calculate the orientation of the forearm with respect to the vertical direction (along the gravity force). This calculation is valid for slow movements in which the total size of the acceleration stays around the value of 1 g. The angles are determined with the following formula:

$$\alpha_i = \arccos\left[\frac{a_i}{\sqrt{a_x^2 + a_y^2 + a_z^2}}\right] \text{ with } i = x, y, z$$

where a_x, a_y, a_z are the 3 components of the acceleration and α_i is the angle between the vertical direction and the acceleration direction under study. In figure 4 these angles are illustrated for the left arm of the singer for three typical postures.

For the study described below the angle in the direction along the forearm is studied (the y-component of the accelerometer). Given the manual annotations of the different postures of the singer the angles for left and right forearm (denoted α_l and α_r respectively) can be determined from the accelerometer data. This leads to the plots shown in figure 5.

One can see that there is a large overlap between posture 1 and 2 which both have similar angles and cannot be disentangled from each other. Furthermore the angular spread

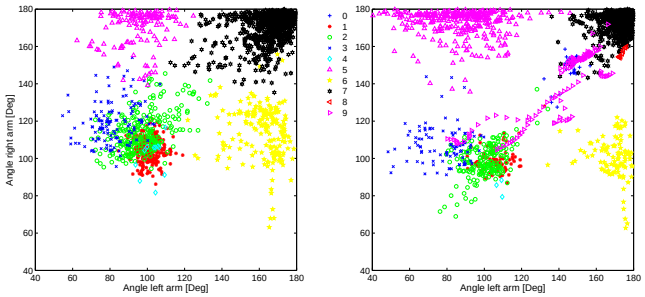


Figure 5. The angle of the left and right forearm extracted from accelerometer data for the rehearsal (left) and concert (right) condition. The different colors/symbols represent the different postures determined from manual annotation

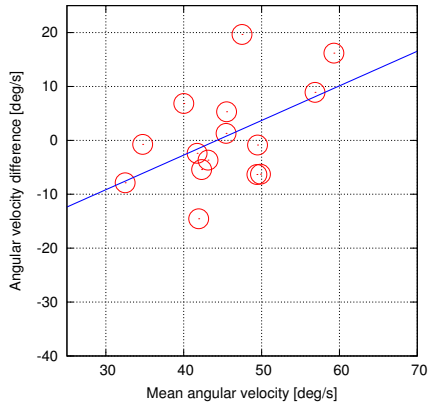


Figure 6. The correlation between the mean angular velocity and the difference in angular velocity for concert and rehearsal conditions. Each data point corresponds to the transitions occurring in a single song of the repertoire

of the data is smaller in the rehearsal condition. This can be quantified by determining the angular center of each posture. The mean Euclidean distance between these centers (in degrees) for the rehearsal and concert conditions are $42.91°$, and $50.72°$, respectively, with a mean difference of $7.81°$. A (single-sided) Wilcoxon signed rank test between the angle pairs corresponding to each posture reveals that the distances between postures are significantly smaller during rehearsal than during the concert condition ($z = -2.5732$, $p = 0.005$).

On the other hand we can also study the transitions between postures based on the calculated angles. When looking at the angular velocity of the transitions occurring in each song a clear correlation is found between the mean velocity of the two conditions and the difference between the velocities as can be seen in figure 6. A linear fit shows a slope of .64. Note that the data point with the highest difference corresponds to the last song which was brought as an encore.

These results are in accordance with previous findings stated in [6] where an increase in intensity of movement was found in the concert condition for songs with a higher average value.

3.3 Automatic recognition of postures during rehearsal and concert

The above analysis of the data is based on manual annotations of postures using video recordings of the performances. This is generally considered to be the most reliable method of annotating data, but even for a moderate amount of data such as used in this study, manual annotation is a very laborious task. However, it is to be expected that each of the postures identified above should have its own signature in the forearm angles as computed from the acceleration data. Indeed, plotting the left and right forearm angles corresponding to the different postures (figure 5) shows various clearly identifiable clusters.

In this subsection we describe a straight-forward method to cluster pairs of forearm angles into postures, and discuss the results. The method consists in first separating postures and transitions, and subsequently clustering the data corresponding postures. In the third part of this subsection we describe the results of the automatic annotation as applied to the data, and evaluate them using the manual annotation.

3.3.1 Separation of postures and transitions

The first step is to determine which time segments correspond to postures and which to transitions from one posture into another. Since a posture is by definition a more or less static position of the body, we use a criterion that states that a time segment represents a posture precisely if the average change of angle per arm does not exceed a particular threshold γ within that segment. This corresponds to the following criterion:

$$\max \left[\begin{array}{l} \frac{1}{(K-1)} \sum_{n=2}^{K} |\alpha_l(n) - \alpha_l(n-1)|, \\ \frac{1}{(K-1)} \sum_{n=2}^{K} |\alpha_r(n) - \alpha_r(n-1)| \end{array} \right] < \gamma \quad (1)$$

where K is the window size, $\alpha_l(n)$, and $\alpha_r(n)$ are the angles of left and right forearm at time n respectively, and γ is the threshold parameter, representing the maximum average change of angle allowed in a posture (in degrees per second).

3.3.2 Clustering of postures

Literature on machine learning and data mining offers a myriad of clustering algorithms. Most, if not all of these algorithms in some form or another rely on parameters to reflect information about the data to be clustered. Depending on the algorithm, parameters can reflect for instance the number of clusters to identify, the maximal variance tolerated within a cluster, or the a priori probability that two data points belong to the same cluster.

In our case, we can easily obtain useful knowledge about the data to be clustered, in the form of approximate forearm angles for the singer's postures. A set of postures is identified by skimming over the video recording, and for each posture an estimation of the forearm angles is made. For example, for the posture 'right hand above left hand' (in front of the body), we expect forearm angles approximately in the middle of the range $180°$ (arm straight down) and $0°$ (arm straight up), where the angle of the right arm is somewhat smaller than that of the left arm. In this manner,

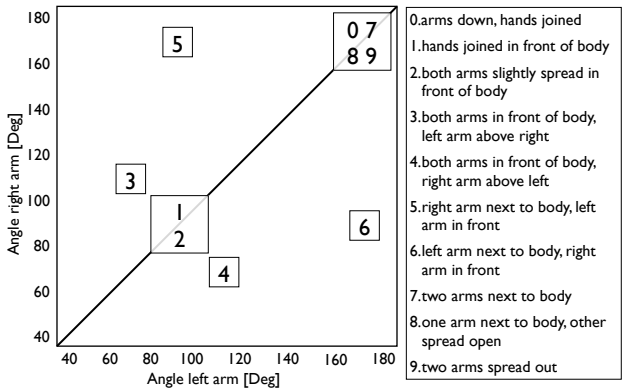


Figure 7. Postures and their typical left and right forearm angle configurations; the x and y axis display the angles of the left and right forearm in degrees, respectively

pairs of forearm angles are defined to serve as cluster prototypes. The prototypes for each posture are displayed in figure 7. Note that the figure contains six postures, rather than ten. The reason for this simplification is that in the limited representation of postures as angles of left and right forearms, some postures are not distinguishable. In particular, postures 1 and 2 (hands joined, and hands slightly open in front of the body), and the various postures where both arms are stretched downward (postures 8, 9, 7, and 0), are all characterized by very similar angles. For this reason, posture 2 has been merged into posture 1 (hands joined), and postures 8, 9, and 0 have been merged into the more frequent posture 7 (both hands down).

Because of the availability of cluster prototypes as a form of knowledge about the data, our particular interest is in clustering algorithms that can take advantage of this knowledge. One such algorithm is the well-known K-means algorithm. Typically, the algorithm is initialized by choosing K random cluster prototypes, but by setting the cluster prototypes deliberately, we can make the algorithm start from hypothesized prototypes, and update these prototypes in accordance with the data.

We also present another simple clustering algorithm that we designed for the purpose of adapting hypothesized cluster prototypes. The algorithm starts from a given set of prototypes, and assigns data points to clusters in a greedy manner, at each occasion allocating the unallocated data point that is closest to any of the prototypes (assigning that point to the prototype that is closest). After the assignment, the prototype of the is updated to reflect the new cluster member.

This clustering method (we will call it *prototype clustering*) is formalized in the pseudo code of algorithm 1. The set X is a set of data points (in this case a vector containing left and right forearm angles) to be clustered using the initial cluster centers (prototypes) c_i ($1 \leq i \leq K$). Furthermore, $X_i \subset X$ denotes the (initially empty) subset of X that belongs to cluster i, σ_i is the variance within cluster i (defined to be zero in case the cluster is empty), and $\overline{X_i}$ denotes the prototype of the set X_i, in this case the centroid of the vectors in X_i.

Algorithm 1: PROTOTYPE-CLUSTERING$(c_1, \cdots, c_K)$

while $X \neq \emptyset$

do
$$\begin{cases} x, i \leftarrow argmin_{x,i} \frac{||x - c_i||}{1+\sigma_i} \\ X_i \leftarrow X_i \cup \{x\} \\ X \leftarrow X / \{x\} \\ c_i \leftarrow \overline{X_i} \end{cases}$$

return $(c_1, \cdots, c_K, X_1, \cdots, X_k)$

The selection of the data point x and and the cluster i to join depends on the variance σ_i of the cluster, in such a way that the agglomeration of data points into disperse clusters is easier than the agglomeration into compact clusters. In this way the criterion for adding a data point to a cluster becomes, informally speaking: 'how much is a data point inside the cloud of data points that form the cluster', rather than 'how far is the data point from the center of the cluster'.

Some other characteristics of this algorithm are that it allows for unequal population of clusters: Clusters do not necessarily get populated, depending on the structure of the data. Furthermore, in spite of its greedy nature, for a given data set, the outcome of the algorithm is robust to variations in initial prototypes.

We have run the *K-means* and the *prototype clustering* algorithms on the posture segments of the forearm angle data, using different values for the threshold parameter γ (eq. 1). The outcome of the various clusterings have been evaluated in terms of precision and recall with respect to the manual posture annotations.

Figure 8 summarizes the results, for the values of γ: 6, 10, 15, 20, 15, 20, 25, 30, 40, 50, and 60 degrees per second. Note that for both clustering algorithms, the threshold values that give optimal accuracies are roughly from 20 to 30 degrees per second. In this range *prototype clustering* slightly outperforms *K-means* clustering. Using $\gamma = 25°/s$, the precision and recall of both algorithms and for both conditions is shown in table 3. Figure 9 shows the corresponding cluster assignments of the angle-pairs graphically. Apart from the differences due to merging the clusters 1 and 2 on the one hand, and 7, 8, 9 and 0 on the other, the resulting clustering is similar to the manual annotations, as displayed in figure 5. Moreover, the effect of postures being more distinct during concert than rehearsal is also observed through the this clustering. This effect was confirmed using a Wilcoxon signed rank test ($z = -4.3493$, $p < 0.005$).

3.3.3 Analysis of transition speeds based on automatic annotation

Apart from the postures themselves, the transitions from one posture to another convey systematic differences between rehearsal and performance, as shown in section 3.2. In particular, songs where the forearm velocity during transitions is low on average (across rehearsal and concert), tend to show lower average forearm velocities during concert than during rehearsal. Conversely, in songs with higher

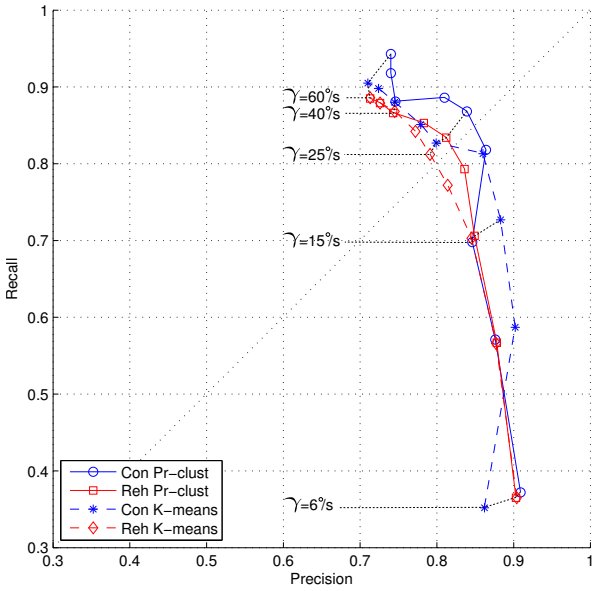


Figure 8. Precision and recall values for the discussed clustering method for concert and rehearsal data sets, using different values of γ (eq. 1)

	Rehearsal		Concert	
	Precision	Recall	Precision	Recall
K-Means	0.79	0.81	0.80	0.83
Pr-Clust	0.81	0.83	0.83	0.87

Table 3. The precision and recall of posture recognition using *K-means* and *prototype clustering* in the concert and rehearsal condition, respectively, using a threshold value of $\gamma = 25°/s$

average forearm velocities, forearm velocities during concert are higher than during concert (see figure 6).

Interestingly, this trend is also clearly visible in the automatic annotation, as shown in figure 10. The slope of the fitted line is 1.13, versus .64 when the analysis is based on manual annotation. Although the slope of the regression line is proportional to the threshold parameter γ (eq. 1), slopes higher than .66 were obtained for all settings of γ. This shows the persistence of the effect, independently of the parameters used for the automatic annotation.

4. DISCUSSION

The results presented in this paper are twofold. Firstly, in subsections 3.1 and 3.2, we have made an analysis of a singer's postures (and transitions) during rehearsal and concerts. This analysis is based on manual annotation of the data. Secondly, in subsection 3.3, we have proposed and evaluated a method to automate the annotation process. In this section, we will discuss the outcome of both parts, starting with the first.

The main findings presented in subsections 3.1 and 3.2 can be summarized as follows: Firstly, with an increase of almost 10%, a greater number of postures was registered during concert than during rehearsal. All postures observed during rehearsal were also observed during concert. Conversely however, three types of postures (8, 9, and 0) observed during concert, were not present during rehearsal. Thus, a wider variety of postures was used by the singer during concert. The open postures 5, 6, and 7 were more frequent during concert, at the expense of posture 1, that can be characterized as a closed position. Furthermore, during concert the transitions between two postures were over 20% longer on average than during rehearsal.

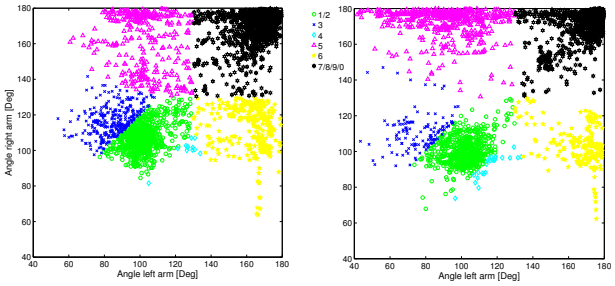


Figure 9. The angle of the left and right forearm extracted from accelerometer data for the rehearsal (left) and concert (right) condition. The different colors/symbols represent the different postures determined from automatic annotation using *prototype clustering* with $\gamma = 25°/s$

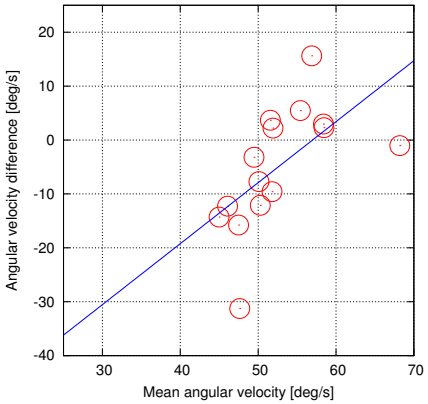


Figure 10. The correlation between the mean angular velocity and the difference in angular velocity for concert and rehearsal conditions. Each data point corresponds to the transitions occurring in a single song of the repertoire

The analysis of the motion sensing data revealed that in terms of left and right forearm angles, postures are less distinct during rehearsal than during concert, in the sense that the clusters representing the various postures (see figure 5) are more separated during concert than during rehearsal. An investigation of the mean angular velocity of transitions per song, as identified using the criterion in equation (1), showed that songs with low average angular velocity tended to have lower angular velocities during concert than during rehearsal, whereas songs with high average angular velocity tended to have higher average velocity during concert than during rehearsal (see figure 6). In other words, the way the singer moved between postures differs from song to song, and these differences are more pronounced during concert than during rehearsal.

From the above findings, it becomes clear that between rehearsal and concert, there are systematic differences in the singer's use of her body as part of the music performance. We assume here that the major difference between the concert and condition is the presence of an audience during the concert, and consequently a musical communication process between performers and audience which is absent during the rehearsal. Based on this assumption, a consistent view arises from the results, which is that the presence of the audience invites, or perhaps even requires the singer to use her body as a means of communication. It leads to a greater variety of postures that are more open in character. Furthermore, during concert the postures are more distinct in terms of the orientation of the forearms, making it easier for the audience to distinguish between different postures. This is in line with previous work on the role of the body in communication, which suggests that an open body position in contrast to a closed body position reinforces the communicator's intent to persuade [7, 8].

The systematic differences in angular velocity during transitions between postures from song to song, and the fact that these differences are amplified during the concert, strongly suggests that transitions between postures also have an expressive function. They may be used by the performer to emphasize changes or other significant moments of the music. The fact that transitions are substantially longer on average during concert can be partly explained by the larger distance between postures. But it can also indicate a stronger emphatic role of transitions during the concert. Further work is needed to investigate this possibility.

It is interesting to see these results in the light of the model of musical communication proposed in [9]. This model consists in the abstract view that the performer encodes his or her musical intentions in sonic and visual energy. This energy is received by the listener [1], who decodes the sonic and visual forms by a mirroring process in order to understand the performer's intentions, in a way similar in nature to the way other social behavior, like empathy, is thought to come about. Such behavior is thought to have a neurological basis [10], in the form of so-called *mirror neurons* (see [11]).

If we assume that the performer's musical intentions stay constant throughout rehearsal and performance, the fact that her corporal behavior is more articulate during the concert might be taken as an intent to encode the musical intentions in a clearer and more detailed way, in order to facilitate disambiguation by the listener in the decoding of the musical intentions.

The results presented in subsection 3.3 show that, with a relatively small amount of prior knowledge (viz. a description of the set of postures in terms of forearm angles), posture annotation can be done automatically based on motion sensing data, rather than manually using video recordings. The accuracy of the annotations in terms of precision and recall lies in the range of 80 to 90 percent. We have also shown that effects observed from manual annotations, such as amplification of differences in average angular velocity during transition can be reproduced using automatic annotations (see figure 6, and 10).

Lastly, a noteworthy result from the automatic clustering of postures based on angular data is that clustering accuracies are generally higher for the concert than for the rehearsal condition, as can be seen from figure 8. This is likely due to the fact that postures are more distinct during the concert, and it illustrates how the singer's efforts to communicate to an audience also facilitate recognition of postures using automatic methods.

[1] In this context 'listener' should be taken more generally to mean observer.

5. CONCLUSIONS

Although it is generally acknowledged that the presence of an audience has an effect on the various types of expression of music performers, not much is known about the way such changes manifest. With the aim of investigating the effect of an audience on the corporal expression of a singer in classical performance together with a gamba player, we have focused on the singer's postures and transitions between postures. Several systematic differences have been found between the dress rehearsal and the concert performances. The findings reinforce the view that the presence of an audience involves a musical communication process between performer and audience that leads to more articulate postures and movements, which are likely to improve the audience's understanding of the performer's musical intentions.

Furthermore, a method was proposed to automate the annotation of postures on which the analysis is based. Rather than manual posture annotation of video recordings, the automated annotation uses motion sensor data. The automatic annotation involves a novel data clustering technique, *prototype clustering*, that can accommodate prior knowledge in the form of cluster prototypes. This technique outperforms the *K-means* clustering algorithm initialized with the same cluster prototypes. An evaluation of this automated annotation method shows that it may be a viable alternative to manual annotation.

6. ACKNOWLEDGMENTS

This work was funded by the Flemish government, under the Methusalem-project *EmcoMetecca*.

7. REFERENCES

[1] A. Camurri, B. Mazzarino, M. Ricchetti, R. Timmers, and G. Volpe, "Multimodal analysis of expressive gesture in music and dance performances," in *Gesture Workshop*, pp. 20–39, 2003.

[2] J. Davidson, "Bodily communication in musical performance," in *Musical communication* (D. Miell, R. MacDonald, and D. J. Hargreaves, eds.), pp. 215–237, New York: Oxford University Press, 2005.

[3] R. I. Godøy and M. Leman, eds., *Musical Gestures: Sound, Movement, and Meaning*. Routledge, 2010.

[4] A. R. Jensenius, M. M. Wanderley, R. I. Godøy, and M. Leman, "Musical gestures: Concepts and methods in research," in *Musical Gestures: Sound, Movement, and Meaning* (R. I. Godøy and M. Leman, eds.), Routledge, 2010.

[5] S. Dahl, F. Bevilacqua, R. Bresin, M. Clayton, L. Leante, I. Poggi, and N. Rasamimanana, "Gestures in performance," in *Musical Gestures: Sound, Movement, and Meaning* (R. I. Godøy and M. Leman, eds.), Routledge, 2010.

[6] D. Moelants, C. F. Wu, M. Demey, and M. Leman, "Performing in concert and in rehearsal: a comparison using audio, video and movement data," *Proceedings of the 2009 European Society for the Cognitive Sciences of Music Conference (ESCOM)*, 2009.

[7] H. McGinley, R. LeFevre, and P. McGinley, "The influence of a communicator's body position on opinion change in others.," *Journal of Personality and Social Psychology*, vol. 31, no. 4, pp. 686–690, 1975.

[8] A. Mehrabian and J. Friar, "Encoding of attitude by a seated communicator via posture and position cues.," *Journal of Consulting and Clinical Psychology*, vol. 33, no. 3, pp. 330–336, 1969.

[9] M. Leman, *Embodied Music Cognition and Mediation Technology*. MIT Press, 2007.

[10] S. D. Preston and F. de Waal, "Empathy: Its ultimate and proximate bases," *Behavioral and Brain Sciences*, vol. 25, pp. 1–72, 2002.

[11] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, "Premotor cortex and the recognition of motor actions," *Cognitive Brain Research*, vol. 3, pp. 131–141, 1996.

EMOTIONS IN THE VOICE: HUMANISING A ROBOTIC VOICE

Tristan Bowles
Department of Theatre, Film and Television
The University of York
York, YO10 5DD, UK
thb500@york.ac.uk

Sandra Pauletto
Department of Theatre, Film and Television
The University of York, UK
York, YO10 5DD, UK
sp148@york.ac.uk

ABSTRACT

The focus of this project is the manipulation of a robotic voice signal for the purpose of adding emotional expression. In particular, the main aim was to design the emotion expressed by a robotic voice by manipulating specific acoustic parameters such as pitch, amplitude and speech rate. The three basic emotions considered were: anger, happiness and sadness.
Knowledge based on the analysis of emotional sentences recorded by actors was used to develop a program in Max/MSP to 'emotionally' manipulate neutral sentences produced by a Text-To-Speech (TTS) synthesiser. A listening test was created to verify the program success in simulating different emotions. We found that test subjects could separate the sad sentences from the others, while the discrimination between angry and happy sentences was not as clear.

1. INTRODUCTION

In many science fiction films we have become accustomed to seeing onscreen robots expressing many human-like emotions. Whether it be the fretful C-3PO (from *Star Wars*), the philosophical Jonny Five (from *Short Circuit)* or the deranged computer HAL (from *2001: A Space Odyssey*), the machines that populate these fictional realities have developed ways of expressing themselves that go beyond their basic programming. The concept of a talking machine is something that is becoming more and more a reality, with the advent of Text-To-Speech systems that allow the user to type text into a computer and have it read back to them by a synthetic voice. The nature of these voices, however, tends to be neutral in tone and often devoid of any signs of emotion and expression.
The intention of this study is to investigate the effect that emotion has on the acoustic signal of the human voice, and apply this knowledge towards replicating three fundamental emotions upon a neutral, synthetic robotic voice. The applications of this research are varied: from the enhancement of speech-based auditory displays, such as Text-To-Speech systems, to speech sound design for creative industries such as film and games.

2. HUMAN EMOTION AND SPEECH

This section introduces fundamental background knowledge on speech production and emotions to facilitate the understating of the project for the reader non-expert in these areas.

2.1 Human speech production

Human speech is produced as air is drawn into the lungs and then pushed through the vocal folds, causing them to vibrate. Depending upon the tension and position of the vocal folds, the varying pressure of air upon the glottis can create a range of different frequencies, which gives 'voice' to the sounds we utter. The sound of our breath can also produce 'unvoiced' sounds, which occur without any vibration of the vocal folds. To illustrate the difference between voiced and unvoiced sounds, Pinker [1, p.164] compares the sounds heard when one makes an unvoiced *sssssss* sound with a *zzzzzzz* sound, which is voiced. Human speech sounds are then further modified as they pass through the mouth.
Humans are also able to change the sound of their voice through intonation. This is achieved through changing the position and tension of the vocal folds, which directly affects the pitch of the voice. As Pinker states, the process of intonation is "what makes natural speech sound different from the speech of robots from old science fiction movies" [1, p.169].

2.2 Definitions of Basic Emotions

Before identifying how emotions affect the human voice it is important to consider what an emotion is. Ekman [2] identifies three important definitions in which commentators have come to understand the term 'basic' emotion. The first definition draws a distinction between 'basic' emotions such as anger, fear, disgust and sadness, believing them to be fundamentally different from one another [ibid., p.45]. The second definition suggests that a 'basic' emotion has evolved from the need to deal with fundamental life tasks, so that they are ways of dealing with situations that are thrown at us during the course of everyday life. Ekman himself believes that "the primary function of emotion is to mobilise the organism to deal quickly with important interpersonal encounters, prepared to do by what types of activity have been adaptive in the past" [ibid., p.46]. A third way of considering a 'basic' emotion is to think of it as the root of some more complex or compounded emotions [ibid., pp.46-7]. Anger, for

instance, could give rise to a hot fury or a cold annoyance.

2.3 Emotional factors that affect the human voice signal

The effect of basic emotions on the human voice is something that can alter the sound of the voice. Indeed, according to Scherer [3], "even slight changes in physiological regulation will produce variations in the acoustic pattern of the speech waveform" [ibid., p.240].

O'Shaughnessy [4, pp.204-242] outlines a number of methods that have been developed for analysing the content of human speech, such as studying the time and frequency-domains and methods for estimating/detecting changes in pitch. Most of the methods have been useful for speech coders who have been trying to re-create human speech synthetically, such as in the Text-To-Speech systems, yet they can also provide analysts with a way of seeing key changes in the voice signal due to factors such as emotions. In the time-domain parameters of the voice, for instance, one can quickly interpret the intensity and rate of speech, as well as the level of the voice [ibid, p.211]. The frequency-domain, on the other hand, can give the speech analyst an idea of the frequencies contained within the voice. In a study looking into identifying personality markers in speech, Scherer [5, p.153] identifies the changes to the fundamental frequency of the voice (in terms of its pitch), the intensity of the voice (how loud certain words are spoken) and the energy distribution within the voice spectrum, which directly affects the quality of the spoken voice, as the main elements that are changed by an emotion experienced by a speaker. Added to this is the further parameter of the rate of speech. This criterion examines the speed in which words are spoken in natural speech, which may include natural pauses, silent periods, and moments where the speech flow is disrupted [ibid., p.160]. So by examining the fundamental frequency, pitch, intensity and speed of the voice, one can begin to identify the changes in the voice that occur when the speaker experiences an emotion.

Scherer [3, p.239] further alludes to studies in which key characteristics of the human voice signal have been digitally changed in an attempt to change the perception of how the voice sounds, and to influence the listener into thinking that the voice has been modified by certain emotions. The main variables of the voice signal that are adjusted in these studies are the fundamental frequency range (F0), the pitch variations or contours, the intensity, duration and accent structure of real utterances [ibid.]. Out of all these variables, Scherer reports that the F0 range had the biggest effect on the listeners, with a narrow F0 suggesting an emotional state such as sadness and a wide F0 "judged as expressing high arousal, producing attributions of strong negative emotions such as annoyance or anger, or for the presence of strongly developed speaker attitudes such as involvement, reproach, or empathic stress" [ibid.]. Similarly, high intensity in the voice signal was perceived negatively, associated with aggressive attitudes, whereas short voiced segments, uttered with a fast speech rate, were interpreted as being joyous in their nature, as opposed to slow speech rate segments with a long duration which were perceived to be akin to emotional sadness [ibid.].

A further factor that may be taken into account when investigating emotion changes in the human voice is to consider the level of emotional arousal that the voice is influenced by. In a recent study into the changes in intonation when emotions are expressed through speech, Bänzinger and Scherer [6, p.257] highlight a distinction between two levels of emotional arousal. High arousal emotions, such as hot anger, panic fear, despaired sadness and elated joy, often are associated with a raised voice, fast speech rate, and higher pitch, when compared to low arousal emotions, such as cold anger, depressed sadness, calm joy/happiness and anxious fear [ibid.]. This division of emotional states into hot and cold elements supports the view held by Ekman [2] that each emotion is not a single emotional state, but belongs to "a family of related states" [ibid., p.55], providing a variety of emotional reactions depending upon the situation in which the emotion is experienced.

2.4 Related studies and novelty of this project

One previous attempt to synthesise speech with emotion is accounted by Murray and Arnott [7], who built a speech synthesiser that modeled six primary emotions: anger, happiness, sadness, fear, disgust and grief. The speech synthesiser attempted to replicate each emotion in four stages. First of all they set some neutral pitch and duration rules that would act as the basis for the synthetic speech before any emotional effects were applied. This acted as a basis for which they developed a second set of rules which attempted to personalise the voice, to give it a certain voice quality, so that, for example, "a breathy voice would remain breathy with emotional effects added later" [ibid., p.371]. Each of the emotion dependent rules were then applied, implementing changes specific to the chosen emotion that the synthesiser was trying to replicate. This included changes such as increasing the pitch and duration of stressed vowels, altering the pitch direction of inflections on word endings, adding pauses after longer words, and eliminating abrupt changes in pitch and between phonemes [ibid., pp.376-7]. The final stage was to send the resulting phonemes and their associated pitch and duration values to the synthesiser, in order to create the speech sound. Murray and Arnott first derive the emotional rules by analysing the emotional speech performed by actors and then verify the success of the rules using a listening test in which subjects are asked to discriminate the emotion portrayed by the synthesised sentences.

Murray and Arnott study [7] is based on sound synthesis or *copy synthesis* [8], i.e. it is a study in which acoustic features are copied from real emotion portrayals and used to resynthesise new emotion portrayals. This type of study is relatively uncommon. Juslin and Laukka made a very comprehensive review of 104 vocal expression studies [8], searching all papers from 1900, and found that only 20% of the studies were based on *copy synthesis*. With this research we hope to add useful knowledge to a research field still relatively unexplored.

An additional novelty of this study is represented by the fact that, instead of synthesising the emotional speech, we

process already synthesised speech samples and "apply" the emotions as an "effect" on the speech sample (rather like a reverb effect is applied onto a dry audio sample). The results of this study should be useful in particular for sound designers with the task of processing speech to make it sound emotional for creative applications (e.g. films and games). It is also important to note that we focus on a non-natural voice, i.e. a robotic sounding voice. For this reason the "naturalness" of the processed voice is not a concern in this study.

3. DESIGNING THE EMOTION CONTENT OF A SPEECH SIGNAL

For this project eight test phrases were chosen to be the basis for emotional analysis and the construction of the program's presets. The phrases were specifically chosen so that they were void of emotional content, i.e. they would not indicate or suggest, from their wording, the emotion being portrayed by the speaker. The 8 phrases consisted of one command statement, "put the butter on the table" (indicated in this paper as phrase 1), two descriptive statements, "the window is wide open" (indicated as phrase 2) and "the orange is round" (indicated as phrase 8), one question, "what time is it?" (indicated as phrase 3), one long statement, "one plus one equals two, two plus two equals four" (indicated as phrase 4), and three phrases containing only monosyllabic or short words: one counting up from 1 to 5 (indicated as phrase 5), one counting down from 5 to 1 (indicated as phrase 6) and one containing 5 numbers in no particular order "one, seven, nine, two, three" (indicated as phrase 7).

3.1 Voice actors

In order to investigate the changes that emotions have on the human voice, four male vocal actors were selected to perform the 8 test phrases, simulating various emotional states. The recording set-up used (equipment, recording studio, and distance between source and microphone) was the same for each performance. The actors were first asked to read out each phrase into an AKG 414 microphone in their "normal" reading voice, thus obtaining a neutral basis through which to compare the proceeding emotional states. The actors were then asked to read out each phrase again in a mildly angry voice, under the direction that they were growing increasingly annoyed at having to read out the phrases. Following this, the actors were asked to read out the phrases in an angry voice once more, but with greater intensity, in order to simulate an explosive anger. These two steps were repeated with the emotions of happiness and sadness, in order to obtain both mild and intense examples of each. It was decided that the intense emotional recordings would act as a basis for the emotional presets in the program to be built for the processing of the robotic voice, mainly because they were the most likely to make a strong impression on the listener.

Once the recordings were made, they were analysed to extract information regarding:

- the exact timing of each word and the length of any pauses or silent periods that appeared within a phrase;
- the fundamental frequency variation and the number of pitch variations or contours per phrase, and whether each pitch contour was upward or downward directed (this was obtained through the pitch analysis pane of the WaveSurfer software [9]);
- the amplitude variation within the phrase.

3.2 Analysis of the actors' voices

In order to analyse the changes that the emotions have on the actors' voices, mean averages of pitch, speech rate and amplitude parameters were taken for each phrase, based upon the data collected from the four actors' speech samples. In this way, the average duration of speech, pitch range, number of pitch contours and amplitude ranges for each of the three emotions were calculated for each phrase. The actors' mean average "neutral" voice was then used as a basis to map the changes to the duration, pitch and amplitude of each phrase.

The test phrases produced a number of common trends that related to type of phrase, the emotion spoken and its effect on the voice.

3.2.1 Phrase duration

Figure 1 shows the average duration of each phrase. Higher bars indicate a longer duration and, when comparing the four emotional versions of the same phrase, a slower speech rate, whereas shorter bars indicate a shorter duration and, when considering different versions of the same phrase, a faster speech rate.

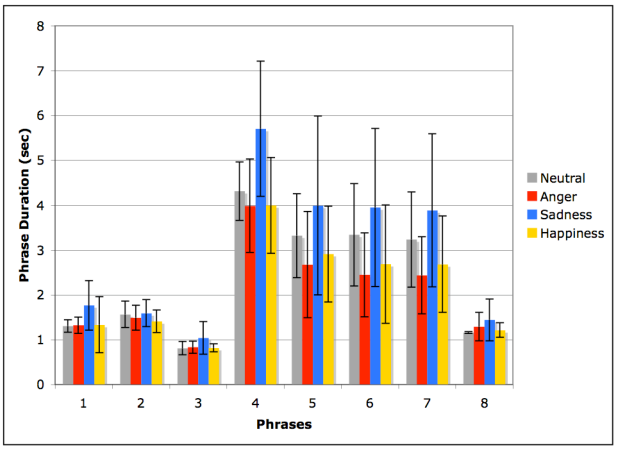


Figure 1: Phrase Duration (averages and standard deviations of the four actors' performances)

We looked at how each emotional phrase deviates from its neutral version and we noted that the phrases involving only monosyllabic or short words (phrases 5, 6 and 7) saw the greatest reduction in duration for the angry and happy phrases (-20% and below the duration of the neutral phrase). In the cases of phrases 1, 3 and 8, the average angry and happy phrases were slower than their equivalent neutral voices. Half of the sad phrases saw over a 20% increase in duration, whereas the short words phrases (phrases 5, 6 and 7) saw an increase in duration between 10-20%. The average length of pauses per phrase was also measured. The sad phrases saw the longest overall pauses with 5 out of the 8 sad phrases having

the longest phrase duration. Half of the happy phrases contained pauses that were longer than the angry equivalent, whereas only angry phrases 2 and 5 contained pauses that were longer than their happy equivalent.

3.2.2 Pitch Analysis

In order to investigate the emotional changes in pitch, the maximum peak in fundamental frequency (F0), the number of pitch contours (or pitch variations) per phrase and the direction of pitch contours were examined.

Figure 2 shows the average maximum peak of F0 for each phrase.

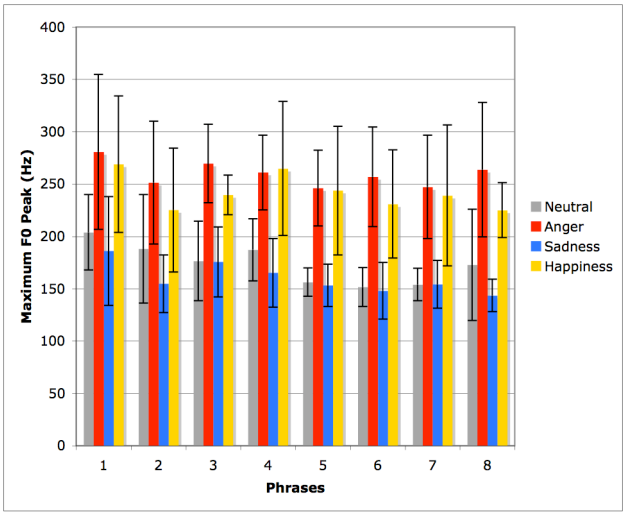


Figure 2: Maximum F0 Peak (averages and standard deviations of the four actors' performances)

For all phrases, anger and happiness have high F0 peaks, while sadness has low F0 peaks.

With the exception of phrase 4, anger phrases have the highest peak fundamental frequency of the 3 emotions. The average maximum peak frequency range of the angry phrases sits between 246Hz-281Hz, with 6 out of the 8 phrases averaging above 250Hz. The happy phrases have a range of 225Hz-269Hz, with 2 of the 8 phrases averaging above 250Hz. The sad phrases have lowest fundamental frequency peaks, operating within a range of 143Hz-186Hz.

Overall, the variation in the number of pitch contours was dependent on the type of phrase. Some of the angry phrases saw the greatest increase in the number of pitch contours, while the happy phrases showed greater variation between increases and decreases, from phrase to phrase. The sad phrases generally saw a decrease in the number of pitch contours, with two exceptions.

The average direction of pitch contours per phrase was calculated by counting every upward curve as a positive value (+1) and every downward directed contour as a negative value (-1). The result for each phrase was totaled and an average obtained. The majority of the neutral phrases contained downward directed pitch contours. The majority of the angry phrases contained more downward directed pitch contours, whereas the happy phrases varied between having upward directed or downward directed pitch contours. The majority of the sad phrases contained downward directed contours.

3.2.3 Amplitude analysis

Figure 3 shows the average maximum amplitude peak for each of the 8 phrases based upon the actors' performances.

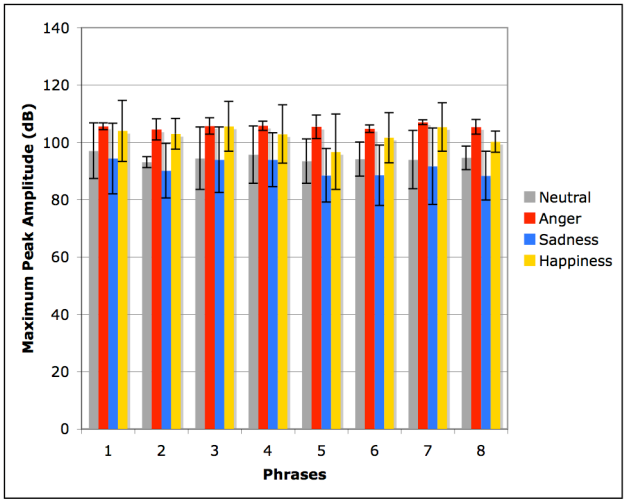


Figure 3: Maximum Amplitude Peak (averages and standard deviations of the four actors' performances)

We can see that for all the phrases, anger and happiness have the high peaks, while sadness have low peaks.

The sad phrases had the lowest maximum amplitude peaks, with all of the sad phrases peaking below 95dB. All the angry phrases and 7 out of 8 happy phrases had peaks that exceeded 100dB.

3.3 The "robotic" text-to-speech voice

In addition to the actors' recordings, the 8 phrases were also recorded by the Text-To-Speech voice "Fred" set to the "Normal" rate of speech, which can be found within the system settings for speech in a Mac computer with the OS X 10.5.8 operating system. This robotic-sounding voice was to act as the primary voice for the program and was to be manipulated to sound angry, sad or happy.

3.4 The Max/MSP program

The emotion-shaping program that was constructed used a real-time granular synthesizer built by Mattijs Kneppers for use in Max MSP 5. Kneppers' [10] patch was chosen because of the way it could smoothly alter the length and pitch of a sound.

Only a few changes were made to Kneppers' original patch in order to make it more suitable for modeling the speech rate, pitch and volume of a speech file over the course of time. Three function graphs were introduced: one controlling the speech rate, one controlling the pitch and one controlling the volume of sound files loaded into the buffer. The ranges for the pitch shift, the time stretch and the volume changes were also adapted to the needs of this project.

3.5 Constructing the Presets

A decision was made to have three presets for each phrase per emotion, so that the total number of presets for each phrase would be nine (3 x angry, 3 x sad and 3 x happy).

The first preset for each phrase was constructed using information collected from the analysis of the actors' readings of the phrases. This was achieved through plotting a graph of the average speech rate for each phrase, and transposing this onto the speech rate graph in the main interface of the Max/MSP program. Average pitch and amplitude graphs of the actors' phrases were also used as a rough guide for the pitch and volume function graphs.

The other two presets were created by tweaking the parameters of the first preset in an attempt to improve the emotional content. This would involve either increasing or decreasing each of the three elements (speech rate/pitch/volume) in order to exaggerate the phrase and make it express a strong emotion. Another method was to listen to specific actors' performances of the phrases and try and match it through the automation of points within the function graphs. In the case of the angry phrases, for example, it was helpful to listen out for the words within the phrase that were exaggerated or stressed the most. The presets that were perceived to be the most successful were then selected for testing purposes.

3.6 Listening test

A listening test was designed to measure the effectiveness of the presets created. The test included the neutral phrases, which were the basis for the emotional presets. This allowed us to verify whether they were really perceived as devoid of emotions.

For the test we adopted a similar approach to Bänzinger and Scherer [6 p.259] who, when testing the changes of intonation in emotional expression, provided their listeners with four visual-analogue rating scales, each representing the "intensity" (ranging from no emotion to extreme emotion) of the four emotions that were the focus of the study.

Similarly we created a test using the 8 neutral TTS phrases and the 24 presets made using the Max/MSP patch (8 x angry, 8 x sad, 8 x happy).

In the test, each participant first entered details pertaining to their date of birth, sex and whether English was their first language, then they were required to listen to each phrase and answer the following two questions:

(i.) *What is the emotional state of the voice?* …to see if their answer matched the intended emotion conveyed through phrase.

(ii.) *How intense is the emotion?* …to gauge how strongly they thought the emotion perceived was expressed.

In order to answer question (i.), the participant had to select one of the four following radio buttons: Angry, Sad, Happy, Neutral. For question (ii.) the participant had to position a horizontal slider between two points one representing weak intensity and the other strong intensity, i.e. the slider ranged from 1 (weak) to 5 (strong). Each participant was allowed to listen to each phrase as may times as they desired, but they had to answer both questions before being allowed to move on to the next phrase. The phrases were played to each participant in a random order so that they were not able to predict what emotion might come next. The test was constructed using the java based audio-visual test-builder *Skatta* [11].

4. LISTENING TEST RESULTS

20 people took part in the test, with an even split between male and female participants. 14 of the test participants were British and had learnt English as their first language, while for the other 6, who varied in ethnicity, English was not their first language.

4.1 Chi-Square Tests Results

The results of the first question of the test were analysed using the non-parametric Chi-Square test. This test can tell us if subjects attributed a particular emotion to a phrase in a random fashion (producing a non-significant result for the Chi-Square, i.e. $p > 0.05$) or not (producing a significant result for the Chi-Square, i.e. $p < 0.05$). Then we looked at which emotion had the most counts and therefore made the bigger contribution towards the significance of the Chi-Square. In the significant cases, we looked at the average score of the intensity of the emotion to gauge how strongly the emotion was portrayed. This is summarised in Table1, where A=Anger, H=Happiness, N=Neutral, S=Sadness, E=significant Emotion and I= Intensity of significant emotion.

Overall: no phrase distinction	**Chi-Square**	**Significance (p)**	**Significant Emotion**	
Anger	72.75	0.000	Happiness	
Happiness	34.35	0.000	Happiness	
Neutral	79.85	0.000	Neutral	
Sadness	222.65	0.000	Sadness	
Phrase by phrase	**Chi-Square**	**Significance (p)**	**E**	**I**
Emotion: Anger				
1-Put the butter…	7.6	0.055		
2-The window…	22.8	0.000	H	3.36
3-What time…	7.9	0.019	H	2.73
4-One plus one…	12.4	0.002	H	3
5-One, two, …	2.8	0.247		
6-Five, four, …	4.0	0.261		
7-One, seven, nine	9.2	0.027	H	2.91
8-The orange …	14.0	0.003	H	2.25
Emotion: Happiness				
1-Put the butter…	4.8	0.187		
2-The window…	6.7	0.035	H	3.17
3-What time…	2.8	0.423		
4-One plus one…	14.8	0.002	H	2.67
5-One, two, …	1.2	0.753		
6-Five, four, …	0.7	0.705		
7-One, seven, nine	2.0	0.572		
8-The orange …	16.3	0.000	H	2.87
Emotion: Neutral				
1-Put the butter…	11.2	0.011	A	3
2-The window…	7.6	0.055		

3-What time…	1.3	0.522		
4-One plus one…	9.2	0.027	N	3.1
5-One, two, …	22.8	0.000	N	3.14
6-Five, four, …	7.2	0.007	N	3.19
7-One, seven, nine	3.7	0.157		
8-The orange …	21.6	0.000	N	2.43
Emotion: Sadness				
1-Put the butter…	11.2	0.011	S	3.09
2-The window…	5.0	0.025	S	2.2
3-What time…	28.9	0.000	S	3.24
4-One plus one…	5.0	0.025	S	3.21
5-One, two, …	7.2	0.007	S	3.63
6-Five, four, …	12.4	0.002	S	3.07
7-One, seven, nine	24.1	0.000	S	4.06
8-The orange …	26.8	0.000	S	3.2

Table 1: Chi-Square results

From these results, we can see that the sad emotion is consistently the best recognised across all phrases. 4 out of the 8 neutral phrases have been correctly recognised, whereas 1 is wrongly recognised as angry. 3 out of 8 happy phrases have been correctly recognised. The angry phrase type failed to be recognised correctly, attaining results of non-significance or significantly happy (in 5 cases). In terms of successful matches per phrase, phrases 4 and 8 have the most number of correctly identified emotions, each sharing an overall success rate of 75%. Phrases 1, 3 and 7 generated the least matches, with only 1 out of the 4 emotions identified correctly.

4.2 Intensity Results

Seven out of eight sad phrases score an intensity higher than 3. The highest intensity score for sadness is 4.06 for phrase 7. Phrase 5 follows with 3.63, then phrases 3, 4, 8, 1, 6, and 2. The highest intensity score for the significantly happy phrases is phrase 2 with a score of 3.17. Phrases 4 and 8 score 2.67 and 2.87 respectively.

Five out of eight angry phrases (2,3,4,7,8) have been incorrectly recognised as happy. Phrases 2 and 4 have scores of 3.63 and 3 respectively, while the rest of the sentences have scores between 2 and 3. It is interesting to note that for these phrases, which were manipulated to portray anger but were recognised as happy, the next emotion that has high scores of intensity is anger indicating that subjects seemed confused between these two emotions.

Four phrases (4,5,6,8) in the neutral case score as significantly neutral, with 3 of these (4,5,6) having intensity scores higher than 3. The highest intensity score is 3.19 for phrase 6. Phrase 1 results as significantly angry with an intensity score of 3. However, as with the confusion between idenification of anger and happy outlined above, the second most chosen emotion is the intended emotion, neutral, with an intensity score of 3.

4.3 Overall confusion matrix

Table 2 shows the confusion matrix for the overall experiment. The numbers represent the percentage of subjects that selected a particular perceived emotion for all the phrases with a specific intended emotion. The Chi-Square is significant for all the emotions.

Intended emotion → / Perceived emotion ↓	**Anger**	**Happiness**	**Neutral**	**Sadness**
Anger	30	11	26	10
Happiness	**49**	**43**	14	2
Neutral	17	25	**53**	13
Sadness	4	21	7	**75**

Table 2: Confusion matrix without distinction of phrase

This overall result shows that sad, neutral and happy phrases are relatively well recognised, while angry phrases are confused as being happy.

5. DISCUSSION

Based upon the results of this study, it is possible to assert that specific changes to the speech rate, pitch and amplitude of the human voice do affect the way the voice is perceived emotionally. However, not all manipulations have been successful in this project. The most evident unsuccessful result is the manipulation to obtain angry phrases. In some cases these have been mistaken for happy.

Less than half of the happy phrases were recognised as happy showing that the approach used is possibly going in the right direction, but requires further refinement.

From the analysis of the actors' voices we can see that anger and happiness have similar trends in terms of speech rate, pitch and amplitude variation. Although anger usually exceeded happiness for the majority of these parameters, there were a few phrases where happiness exceeded anger. This seems to indicate a close relationship between the emotions of anger and happiness that makes it a lot more difficult to distinguish between the two when attempting to simulate them onto a robotic-sounding voice. One possible reason why the angry and happy phrases obtain poor results could be that further stimuli, such as a facial expression or an emotionally-loaded phrase, is needed for a participant to be able to distinguish between an angry voice and a happy voice. Alternatively, maybe further alterations to the voice, that go beyond simply adjusting the speech rate, pitch and amplitude, need to occur before anger becomes more distinct from happiness.

The successful recognition rate for the sad phrases confirms that if the pitch, duration and amplitude of the robotic neutral voice all decrease, then the voice will sound sad.

Four out of eight neutral phrases were recognised as neutral. For three neutral phrases no clear emotional state could be assigned therefore the "neutralness" of these phrases is unclear in this test. It is possible that with a larger number of subjects the perceived emotional state of these sentences could emerge more clearly.

In the case of neutral phrase 1, the participants rated the emotion as being angry. A possible reason for this may be the words within the phrase. Whilst the phrase itself doesn't contain any emotion-specific words, the nature of the sentence is a command. Participants may have thought this sounded forceful and angry in this instance. Murray and Arnott [7] allude to a similar occurrence when testing the phrases from their HAMLET system, where the words within the phrase itself caused participants to imagine "a stereotypical situation where the phrase might be used" [ibid., p.387].

Overall, this project is successful in giving more insight into methods that could be employed to design emotional expression in a robotic voice. There are, however, a number of improvements that can be made to the work.

Difficulties were encountered when constructing the presets. This related to how the pitch and amplitude data, based on the actors' voices, could be used accurately in the program. As the pitch and volume of the actors' voices vary significantly over the course of a phrase, it was particularly difficult to judge on which words to make specific pitch and amplitude adjustments. Indeed, one might benefit from trying to model the pitch and amplitude of a robotic voice based on one actor's performance, rather than an average of several.

Currently, the function graphs in the user interface of the Max/MSP patch used to manipulate the phrases can be used to sketch out the changes in speech rate/pitch/amplitude over the course of a phrase with relative ease, but at the cost of accuracy.

Furthermore, the presets used for testing could be improved by gaining feedback from a panel of experts (e.g. theatre directors, actors, etc.).

A further, larger study that includes the improvements mentioned above has been planned and is under development at present. We plan to report the results of this new experiment in the near future.

6. CONCLUSION

A program was constructed during this study that "applies" three emotional states onto a robotic-sounding Text-To-Speech voice. This study has examined how changes in speech rate, pitch and amplitude affect the human voice and has applied this knowledge towards creating a number of emotional presets that allow the user to make a robotic speech signal sound angry, sad or happy. The resulting presets were tested on a wider audience and the results from this test indicated that decreases in speech rate, pitch and amplitude create the impression of sadness. The test also showed that the angry and happiness presets were very closely related, indicating that speech rate, pitch and amplitude adjustments alone, or without a greater degree of accuracy, are not enough to make anger distinct from happiness. From these results, a number of enhancements and further work was suggested, which should help to clarify how we can improve emotion expression in a robotic voice.

7. REFERENCES

[1] S. Pinker: *The Language Instinct*, St Ives: Penguin Books, pp.158-191, 1995.

[2] P. Ekman: "Basic Emotions", pp. 45-60, in T. Dalgleish and M. Power: *Handbook of Cognition and Emotion*, Chichester: John Wiley & Sons, 1999.

[3] K. R. Scherer: "Expression of emotion in voice and music", *Journal of Voice* 9(3): 235-248, 1995.

[4] D. O'Shaughnessy: *Speech Communication: Human and Machine*, United States: Addison-Wesley Publishing Company, pp. 204-242, 1987.

[5] K. Scherer: "Personality markers in speech", in K. Scherer and H. Giles (eds.): *Social Markers in Speech* London: Cambridge University Press, pp. 147-210, 1979.

[6] T. Bänzinger and K. Scherer: "The role of intonation in emotional expressions", *Speech Communication*, vol 46, pp.252-267, 2005.

[7] I. Murray and J. Arnott: "Implementation and testing of a system for producing emotion-by-rule in synthetic speech", *Speech Communication*, vol.16, issue 4, pp.369-390, 1995.

[8] P.N. Juslin and P. Laukka: "Communication of Emotions in Vocal Expression and Music Performance: Different Channels, Same Code?", *Psychological Bulletin*, Vol. 129, No. 5, 770 – 814, 2003.

[9] Wavesurfer Software: available from <http://www.speech.kth.se/wavesurfer/> [Last Accessed 14 June 2010].

[10] M. Kneppers: "Real-time, natural sounding granular time stretcher/ pitch shifter", available from <http://www.cycling74.com/share.html> [Last Accessed 14 June 2010].

[11] Skatta Software: available from <http://sourceforge.net/projects/skatta/> [Last Accessed 14 June 2010]

REAL-TIME ESTIMATION OF THE VOCAL TRACT SHAPE FOR MUSICAL CONTROL

Adam P. Kestian and Tamara Smyth
School of Computing Science, Simon Fraser University
apk6@sfu.ca, tamaras@cs.sfu.ca

ABSTRACT

Voiced vowel production in human speech depends both on oscillation of the vocal folds and on the vocal tract shape, the latter contributing to the appearance of formants in the spectrum of the speech signal. Many speech synthesis models use a feed-forward source-filter model, where the magnitude frequency response of the vocal tract is approximated with sufficient accuracy by the spectral envelope of the speech signal. In this research, a method is presented for real-time estimation of the vocal tract area function from the recorded voice by matching spectral formants to those in the output spectra of a piecewise cylindrical waveguide model having various configurations of cross-sectional area. When a match is found, the formants are placed into streams so their movement may be tracked over time and unintended action such as dropped formants or the wavering of an untrained voice may be accounted for. A parameter is made available to adjust the algorithm's sensitivity to change in the produced sound: sensitivity can be reduced for novice users and later increased for estimation of more subtle nuances.

1 INTRODUCTION

Estimation of the vocal tract area function from an incoming voice signal is a task that has numerous proposed solutions, as several applications would greatly benefit from accurate depictions of the vocal tract shape during speech. For example, studies have found that displaying the vocal tract to the hearing impaired can improve their overall speech performance [11, 15] as well as being an effective instructional tool for singers [14]. Recent studies have aimed to identify useful features from the voice signal for musical control [7, 6]. Some have identified the vocal tract shape as a potential feature for musical control and have used vision-based methods for its estimation [5, 10]. Here, we propose using the vocal tract shape as input data for control, though leaving the actual application and mapping strategy to the user, and present a method for extracting this shape directly from recorded voice by calibrating to the output of a waveguide model.

Human speech depends both on the oscillation of the vocal folds in the glottis and on the shape of the vocal tract. Producing a particular vowel sound requires changing the effective cross-sectional area function of the vocal tract that contributes to the appearance of formant peaks in the frequency spectrum of the speech signal. While several algorithms have been proposed to identify the vocal tract area function from recorded speech, the most common involve either computing reflection coefficients from linear predictive coding (LPC) or autoregressive (AR) models using, for example, Yule-Walker or Levinson-Durbin methods [2, 18], or by directly tracking formant peaks in the spectral envelope of recorded speech [9, 3]. Both approaches assume a simplified feed-forward source-filter model of the voice, and produce best results on vowel sounds, as the articulation of consonants is more complicated than purely changing the vocal tract shape.

For many applications, the feed-forward filter model is considered to be sufficiently accurate as there is weak coupling between the massy vocal folds and the vocal tract. It should be mentioned however, that a stronger influence of the vibrating source is observed on the spectral envelope—and thus the formant peaks—for feed-back models that more strongly couple the glottal excitation and vocal tract [17]. Likely due to this and other simplifications in the filter model, such as inaccurate estimates of unknown propagation losses and termination reflection/transmission functions, the detection of vocal tract shape from reflection coefficients has produced inconsistent and inaccurate results, both in this work and in that of other authors [9, 12, 19]. Furthermore, accuracy deteriorates rapidly with an increase in sampling rate, making its use limited for the singing voice which typically requires a greater bandwidth than its spoken counterpart.

Formant-based analysis-by-synthesis methods may produce better results as they overcome filter inaccuracies by only requiring a most likely fit to a synthesis model's output. Depending on the method used, calibrating a recorded signal to the output of a very detailed and accurate model could introduce potentially restrictive computational cost, thus making it inappropriate for real-time use. In this particular case, there is also the possibility of estimating one of several vocal tract shapes that produce indistinguishable

spectra. Care must therefore be taken to constrain possible shapes so they are physiologically reasonable.

In this work, we present an efficient and accurate formant-based vocal tract area function estimation algorithm, specifically designed for real-time musical control. As described in Section 3, the algorithm identifies formants on a sample frame basis, and places them into streams, enabling their movement to be tracked over time.

In Section 4, minimum action is applied to improve usability, algorithm performance, and the visual feedback to the user, by smoothing formant streams to account for unintended action such as dropped formants or the wavering of an untrained voice (see Section 4). It considers that some users will have greater control of their voice than others, and provides a parameter for adjusting the algorithm's sensitivity to a change in the produced sound.

Section 5 describes how estimated formants are compared and matched to a database of formants collected offline from the output of the vocal tract model (described in Section 2) having various configurations of cross-sectional area.

2 A SIMPLIFIED VOCAL TRACT MODEL

It is well known that digital waveguide modeling may be used to simulate plane and spherical waves propagating in cylindrical and conical acoustic tubes [16]. More complex shapes, such as those produced by the vocal tract when using the velum, jaw, tongue and lips to voice different vowel sounds, can be approximated using a sequence of cylindrical waveguide elements separated by scattering junctions accounting for the reflection and transmission that occurs when a change in cross-sectional area creates a corresponding change of impedance.

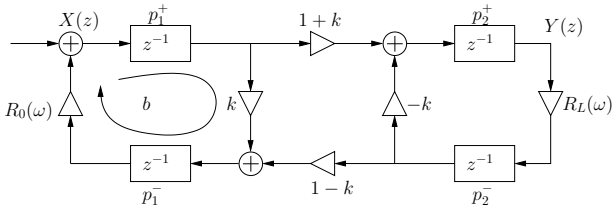


Figure 1. A waveguide model of an acoustic tube closed at one end with reflection $R_0(\omega)$ and open at the other with reflection $R_L(\omega)$. The tube has a single change in cross-sectional area at the center, creating a two-port scattering junction with reflection and transmission defined by k, a coefficient set according to the relative areas of the two sections.

The way in which a shape departing from the purely cylindrical contributes to formant peaks in the magnitude spectrum may be seen by considering a simple model with only two cylindrical waveguide segments of the same length but with different cross-sectional areas, A_1 and A_2 respectively (see Figure 1). The two-port scattering junction models the reflection and transmission that occurs at the change in cross-sectional area, where the reflection coefficient is given by

$$k = (A_1 - A_2)/(A_1 + A_2). \tag{1}$$

The response at $Y(z)$ (corresponding to the mouth) to input $X(z)$ (corresponding to the glottis) is given by

$$\begin{aligned} Y(z) &= X(z)(1+k)z^{-2}[1+b+b^2+\ldots] + \\ &\quad Y(z)R_L(z)R_0(z)(1-k^2)z^{-4}[1+b+\ldots] + \\ &\quad Y(z)R_L(-k)z^{-2}, \end{aligned} \tag{2}$$

where

$$b = R_0(z)kz^{-2}. \tag{3}$$

Equation (2) is the sum of three terms corresponding to the possible signal flow paths to $Y(z)$, with the first two terms including the infinite series generated by the circulating path in the first waveguide segment. Using the closed form resprepresentation and substituting (3) into (2) yields all-pole filter transfer function

$$\begin{aligned} H(z) &= Y(z)/X(z) \\ &= \frac{(1+k)z^{-2}}{1+k(R_L(z)-R_0(z))z^{-2}-R_L(z)R_0(z)z^{-4}}. \end{aligned} \tag{4}$$

Setting end reflection functions

$$R_0(z) = 1, \quad \text{and} \quad R_L(z) = -1$$

makes the system lossless with transfer function

$$\hat{H}(z) = \frac{(1+k)z^{-2}}{1-2kz^{-2}+z^{-4}}, \tag{5}$$

preserves the harmonic structure of an open end tube, and allows for observation of the effects of the junction.

Factoring the denominator in (5) yields the intermediate complex conjugate pair,

$$\rho = k + j\sqrt{1-k^2} \qquad \rho^* = k - j\sqrt{1-k^2}, \tag{6}$$

and ultimately the four roots/poles of the filter given by

$$p_1 = \sqrt{\rho}, \quad p_2 = -\sqrt{\rho}, \quad p_3 = p_1^*, \quad p_4 = p_2^*. \tag{7}$$

Figure 2 shows how the poles shift as a function of k and thus in response to a change in cross-sectional area. Shifting poles corresponds to a shift of harmonic peaks in the magnitude which, when more sections with varying cross-section are considered, leads to regions in the spectrum with increased and decreased energy, and the appearance of formant peaks during vowel production.

Increasing the number of segments increases the number of poles in the vocal tract transfer function. As shown in (6) and (7), filter poles are a function of the reflection

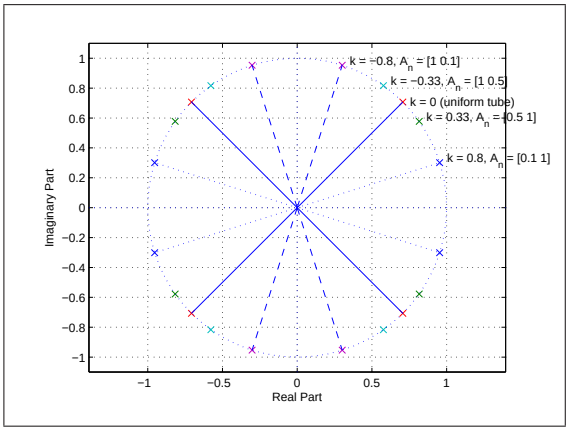


Figure 2. The four poles of transfer function (5) plotted for five values of reflection coefficient k. The uniform cylindrical tube has a reflection coefficient of $k = 0$ and corresponds to a uniform/harmonic spacing of poles (or peaks in the magnitude spectrum). A change in k corresponds to a change in the cross-sectional area to the tube and the observed shifting of poles in the vocal tract transfer function.

coefficient, allowing a change in cross-sectional area to be inferred directly from filter poles. The complexity involved in this recursive problem however, is unnecessarily expensive (though not prohibitively so) for real-time applications, and yields far more data than is required to identify the vocal tract shape with the accuracy desired here. Rather, it was found that a vocal tract shape could be sufficiently characterized using only up four formant peaks in the magnitude of its frequency response.

3 TRACKING FORMANTS IN VOCAL OUTPUT

An incoming sample frame of recorded voice is windowed and processed to extract its spectral envelope—a curve assumed to approximate the magnitude of the vocal tract frequency response—with formants peaks in the spectra being identified by tracking curve local maxima.

Notice from the log spectrum in Figure 3 (upper curve) that the position of weaker formants is sometimes obscured by the presence of more pronounced formants having greater amplitude and bandwidth. As shown by the broken-line curve in Figure 3 (lower curve), the second derivative of the log magnitude spectrum may be taken to produce more prominent bends in the curve contour at peak locations, effectively reducing the formant bandwidth and accentuating the position of "merged" formants [13]. Though is also possible to take the third derivative of the phase spectrum [8] to yield further improvement in bandwidth attenuation and peak accentuation, this method was found to be less successful in tracking merged formants with significantly different amplitudes, and thus produced less consistent results.

Once the most prominent formant peaks are detected,

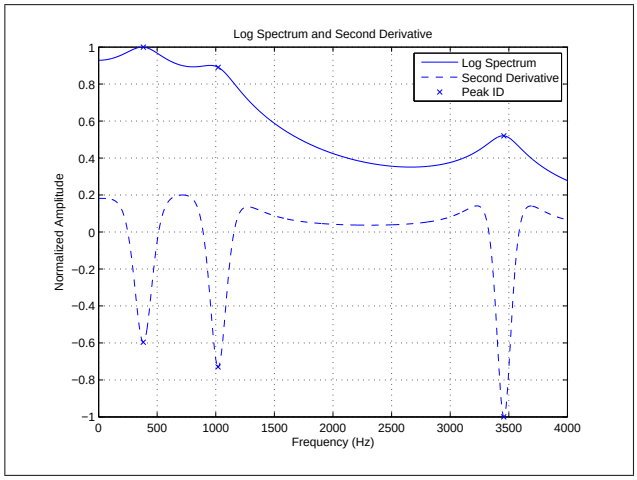


Figure 3. The log magnitude spectrum of an input sample frame (upper solid line) and its second derivative (lower broken line), with the latter accentuating the position of "merged" formants.

they are placed into formant streams that track the movement of a formant number from frame to frame. These formant streams are necessary to account for dropped formants and improve usability and performance as discussed in Section 4. Limiting the streams to a maximum of four was sufficient to uniquely identify a corresponding vocal tract model.

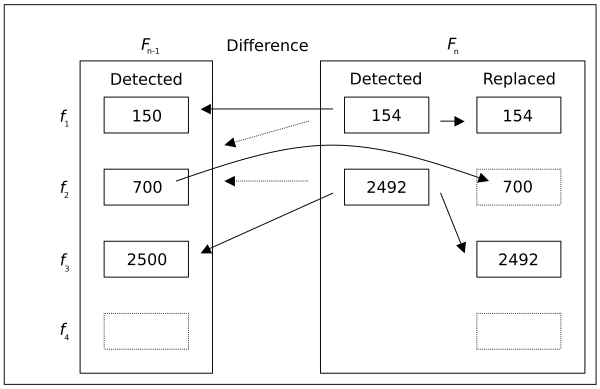


Figure 4. Example of formant stream assignment: Analysis of current frame F_n yields two detected formant peaks at 154 Hz and 2492 Hz. The first peak at 154 Hz is closest to the first formant of the previous frame F_{n-1} and is thus assigned to the first formant stream $f_1(n)$. Similarly, the peak at 2492 Hz is assigned to the third formant stream $f_3(n)$. To accommodate for the "dropped" formant in the second stream, the last value assigned from the previous frame is held over to the current frame.

To determine to which stream a formant peak should be assigned, a distance measure is taken between the estimated formant peak of a current frame F_n and the stream-assigned neighbouring formants of the previous frame, F_{n-1}, with

the formant being assigned to the stream of its neighbour closest in frequency. If the difference between formant frequencies exceeds a threshold, an additional formant stream becomes active. Though it is possible to have up to four streams, it is most common to use only three.

The process is illustrated by the example in Figure 4, where only two peaks have been detected with three active streams, flagging the possibility that a formant was unintentionally "dropped". Accounting for such absent formant peaks, as described in Section 4, further improves usability of the system.

4 MINIMUM ACTION FOR IMPROVED USABILITY

There are the two situations in which a formant may unintentionally temporarily disappear from one sample frame to the next: 1) when the algorithm fails to detect it for a particular frame or more likely 2) an untrained voice is unable to consistently sustain the quality of the produced sound. As shown in Figure 5 (top), either instance generates a sharp null in the formant tracking curve.

To accommodate for this, minimum action is assumed, and such significant temporary drops in the curve are considered unlikely or unintentional (with minimum action suggesting too much effort would be required to intentionally produce such a drastic change). Consider again the example shown in Figure 4, which shows only two peaks being detected in the presence of three active formant curves. Since the detected peak is placed in the third stream, it is the second formant that was dropped. In this case, the last value in the second stream is held over to the current frame, $f_2(n) = f_2(n-1)$. In this way, when/if the formant returns in subsequent frame analysis, it will be placed in the correct stream and the sharp nulls in the curves will be avoided (see top and middle of Figure 5). The repetition of formants within a stream can occur up to four times before the stream is deemed inactive.

Algorithm performance and visual feedback to the user is further improved by applying a smoothing filter to the formant tracking curves, effectively stabilizing the movement of the formants and compensating for unintentional wavering of the less-trained voice. An amplitude envelope follower given by,

$$\hat{f}_m(n) = (1-\nu)|f_m(n)| + \nu\hat{f}_m(n-1), \qquad (8)$$

is applied to the formant streams $f_m(n)$, where ν determines how quickly changes in $f_m(n)$ are tracked. If ν is close to one, changes are tracked slowly, making the smoothed curve $\hat{f}_m(n)$ less sensitive to change. If ν is close to zero, $f_m(n)$ has an immediate influence on $\hat{f}_m(n)$. A higher ν, therefore may be appropriate for untrained voices, but with practice, the parameter value may be decreased to allow for better detection of subtle nuances.

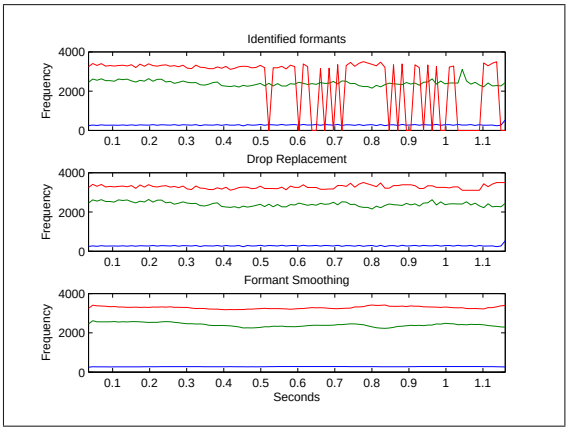


Figure 5. Three active formant streams with the third stream having sharp nulls due to temporarily dropped formants (top). Nulls are avoided by holding values from the previous frame when a formant is flagged as being dropped (middle). Further smoothing is applied to compensate for unintentional wavering in the less-trained voice (bottom).

Regardless of ability, formants shift rapidly during an onset of (or change in) the vocal/vowel sound, and thus detected formants are added to streams only once the formants have settled and the speech waveform is more sustained. (This creates latency in the visual feedback to the user equal to the onset duration). Extensive methods for detecting attacks in sounds from musical instruments are not necessary here, particularly since they are considered to be less effective when applied to the voice [4]. Figure 6 shows the waveform recorded when a speaker produces the vowel sounds /ee/ to /oo/ and back to /ee/. In spite of the speaker's attempt to keep the amplitude constant, the waveform envelope clearly shows a change in energy at the locations of the changing events. This result is expected when considering that the waxing and waning in the frequency spectrum due to shifting formants will have an equivalent effect on the signal's energy in both time and frequency domains (Parseval's theorem).

The onset of an event is therefore identified solely by tracking steep slopes in the amplitude envelope of the speech signal. At an event onset, the formant peaks are still detected, but with their rate of change being recorded from frame to frame. Once this value is sufficiently reduced and the position of the speaker/singer's formants settle, the onset region is considered to have passed and the algorithm may resume with the process of formant stream assignment and vocal tract shape estimation.

5 ESTIMATION OF THE VOCAL TRACT SHAPE

With the estimated formant streams in place, the final step is to search the database produced by the output of the model configured to various shapes, and find the most likely can-

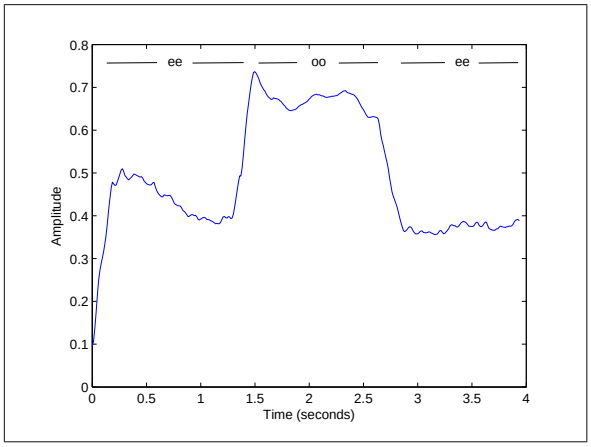


Figure 6. Amplitude envelope produced when voicing vowels /ee/ then /oo/ then back to /ee/ while attempting to keep the amplitude (loudness) constant. The amplitude envelope clearly shows the locations of these event changes.

didate.

A piecewise cylindrical waveguide model, similar to that shown in Figure 1, was developed having N_s sections, each with N_A possible cross-sectional area values, yielding $N_A^{N_s}$ possible combinations. Considering all possible combinations yields duplicate shapes however, fewer corresponding vocal tract area functions are produced by considering only the change in cross section. Here, $N_s = 7$ and $N_A = 5$ seemed to produce the best results when considering performance, usability and accuracy, and the intended application of real-time musical control.

A table maps the model's vocal tract area function to the formant frequencies in its output spectra. The table is sorted by grouping the shapes based on the number of detected formants. The formants detected from the incoming speech signal (as described in Section 3) are compared to the entire portion of the database having the same number of formants. The Euclidean distance $d(fU, fM)$ is used to measure which set of formants generated by the model fM_m best match those generated by the user fU_m:

$$d(fU, fM) = \sqrt{\sum_{m=1}^{M} (fU_m - fM_m)^2}, \qquad (9)$$

where M is the number of detected formants. The estimated vocal tract shape displayed to the user remains static until another event onset is detected. Once a new shape is identified, linear interpolation is performed over the next frame to smooth the transition, and thus the visual display, between the two shapes.

6 RESULTS AND CONCLUSIONS

Figures 7 and 8 provide vocal tract estimation examples of sung vowels /au/ and /ee/. The lower plots of the two fig-

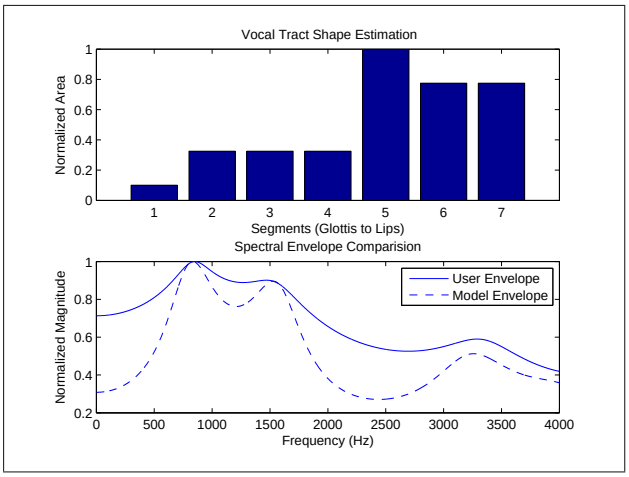


Figure 7. Estimation of vocal tract shape for vowel sound /au/.

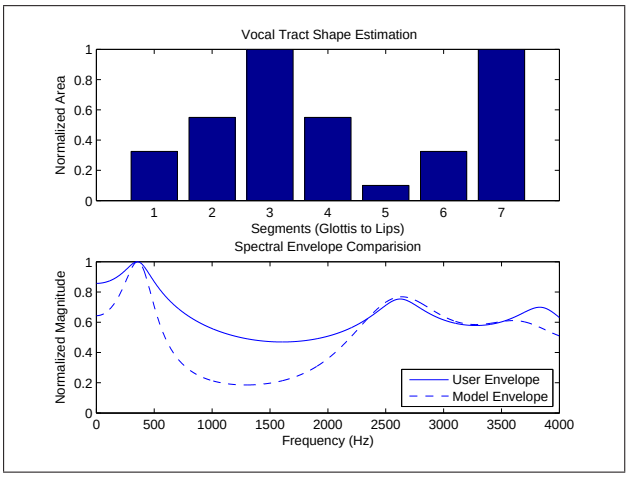


Figure 8. Estimation of vocal tract shape for vowel sound /ee/.

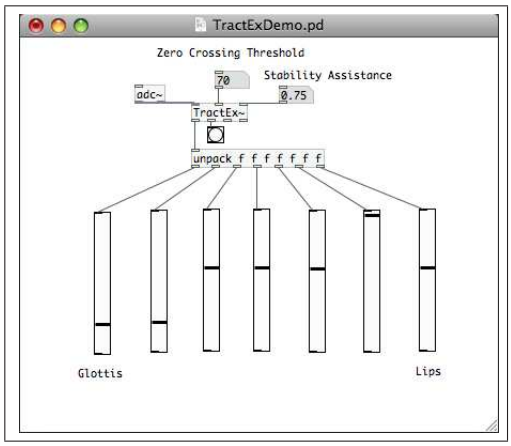


Figure 9. Pure Data object displaying /au/ example

ures compare the spectral envelope of the voice signal to the spectral envelope generated by the vocal tract model of the estimated shape. The differences between the spectral envelope of the voice and the model can be attributed to simplifications of the source-filter model, the wave propagation losses and unknown termination reflection/transmission functions (as discussed in Section 1). Also a factor is the limited resolution of the vocal tract model look-up table. Though increasing the resolution would produce more spectral envelopes for comparison, it would increase the number of permutations and possibly adversely affect real-time performance without necessarily improving estimate accuracy.

With this method, we provide a vocal tract area function estimation algorithm that offers a suitable level of sensitivity for users having varying abilities. The use of formant streams enable a formant's movement to be tracked over time so that the vocal tract shape may be stabilized by accounting for unintended action, thereby improving its use for musical control.

Strategies for mapping the vocal tract area function data to control other music applications are left to the user. The authors are currently interfacing this work to the control of parametric synthesis models and polyphonic virtual instruments.

The algorithm is implemented in PureData (Pd) [1], a real-time audio programming environment popular among musicians. This facilitates control data acquisition, visual and audio feedback display, as well as mappings to other music and sound synthesis applications. The object will be made available to the public upon request.

7 REFERENCES

[1] "Pd," http://www.pure-data.org.

[2] P. Broersen, "Finite sample criteria for autoregressive order selection," in *IEEE Trans. Signal Processing*, vol. 48, 2000, pp. 3550–3558.

[3] J. Dang and K. Honda, "Estimation of vocal tract shapes from speech sounds with a physiological articulatory model," in *Journal of Phonetics*, vol. 30, 2002, pp. 511–532.

[4] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006, pp. 133–137.

[5] V. Florian, G. McCaig, M. Ali, and S. Feis, "Tongue 'n' groove: an ultrasound based music controller," in *Proc. New Interfaces for Musical Expression*, 2002.

[6] A. Hazan, "Performing expressive rhythms with billaboop voice-driven drum generator," in *Proc. of the 8th Int. Conference on Digital Audio Effects*, 2005.

[7] J. Janer and M. de Boer, "Extending voice-driven synthesis to audio mosaicing," in *5th Sound and Music Computing Conference*, 2008.

[8] C. Jinhai, J. Gangji, and Z. Lihe, "A new method for extracting speech formants using lpc phase spectrum," in *IEEE Electronics Letters*, vol. 29, 1993, pp. 2081–2082.

[9] P. Ladefoged, R. Harshman, L. Goldstein, and L. Rice, "Generating vocal tract shapes from formant frequencies," in *J. Acoust. Soc. Am*, vol. 64, 1978, pp. 1027–1035.

[10] M. Lyons, M. Haehnel, and N. Tetsutani, "Designing, playing, and performing with a vision-based mouth interface," in *Proc. New Interfaces for Musical Expression*, 2003.

[11] S. Park, D. Kim, J. Lee, and T. Yoon, "Integrated speech training systems for hearing impaired," in *IEEE Transactions on Rehabilitation Engineering*, vol. 2, 1994, pp. 189–196.

[12] D. Paul, "Estimation of the vocal tract shape from the acoustic waveform," in *Ph.D. Diss., Massachusetts Inst. Technol.*, 1976.

[13] P. P. R. Christensen, W. Strong, "A comparison of three methods of extracting resonance information from predictor-coefficient coded speech," in *IEEE Trans. Acoust. Speech Signal Process.*, vol. 24, no. 1, 1976, p. 814.

[14] D. Rossiter, D. Howard, and M. Downes, "A real-time lpc-based vocal tract area display for voice development," in *Journal of Voice*, vol. 8, 1994, pp. 314–319.

[15] M. Shah and P. Pandley, "Areagram display for investigating the estimation of vocal tract shape for a speech training aid," in *Proc. Symposium on Frontiers of Research on Speech and Music*, 2003, pp. 121–124.

[16] J. O. Smith, *Digital Waveguide Modeling of Musical Instruments*. ccrma.stanford.edu/˜jos/waveguide/, 2003, last viewed 12/4/08.

[17] T. Smyth and A. Fathi, "Voice synthesis using the generalized pressure controlled-valve," in *Proceedings of ICMC 2008*, Belfast, Ireland, August 2008, pp. 57–60.

[18] H. Wakita, "Direct estimation of the vocal-tract shape by inverse filtering of acoustic speech waveforms," in *IEEE Trans. Audio Electroacout.*, vol. AU-21, 1973, pp. 417–427.

[19] ——, "Estimation of vocal-tract shapes from acoustical analysis of the speech wave: the state of the art," in *IEEE Trans. Acoust. Speech Signal Process*, vol. ASSP-27, 1979, pp. 281–285.

CREATION AND EXPLORATION OF A PERCEPTUAL SONIC TEXTURES SPACE USING A TANGIBLE INTERFACE

Jean-Julien Filatriau
Communications Lab (UCL-TELE)
Université catholique de Louvain, Belgium

Daniel Arfib
Multicom-LIG
Grenoble, France

ABSTRACT

This study takes place in the framework of an ongoing research dealing with the analysis, synthesis and gestural control of sonic textures. Sonic textures include a wide range of sounds featured by random properties in a short-term scale and constant characteristics in a long-term scale. In this paper, we describe two recent contributions related to this field: the first one aimed at providing a sonic textures space based on human perception. For that purpose, we conducted a psychoacoustic experiment, relying on a tangible interface, where subjects were asked to evaluate similarity between sonic textures by gathering them in several groups. The second part of this study aimed at experimenting the control of sonic textures synthesis using a tangible interactive table. We also designed a musical tabletop application inspired by the metaphor of a sonic space exploration. This gave very promising insights on the possibilities offered by such interfaces for the real-time processing of sonic textures.

1. INTRODUCTION

Before presenting the two main contributions of this paper, i.e. the creation of a perceptual space of sonic textures relying on the result of a psychoacoustic experiment and its exploration by an interactive tangible interface, we will introduce in the following sections some important concepts underlying this research: first we will bring some elements defining the class of the sonic textures, then we will compare the two approaches that can be adopted when addressing the question of sound classification and finally present the original RFID-based tangible interface we used for this study.

1.1 The sonic textures

Sonic textures are characterized in terms of both microscopic and macroscopic features: on the short-term scale, they are composed of a series of micro-structural elements which are subject to randomness, whereas on the long-term scale, the characteristics of the structure and randomness remain constant. Some environmental sounds, such as rain, waterfalls or wind, are amongst the most obvious examples of this group of sounds but it is also possible to find textures in a musical context, especially in contemporary or electro-acoustic compositions [1]. The computer music community has recently drawn attention to sonic textures [2] and both synthesis and analysis-synthesis methods, specifically dedicated to this class of sounds, have been developed lately [3, 4].

1.2 Features-based vs. perceptual classification of sounds

Two main approaches can be followed when dealing with the characterization and classification of musical sounds. The first one refers to Music Information Retrieval (MIR) and relies on an objective description of the sound based on a series of features computed on the audio signal [5, 6]. The MIR community enhanced quickly over the last decade and a number of MIR tools, such as Marsyas [1], CLAM [2] or the MIRToolbox [3], are hence available nowadays for experimentation.

The second strategy, which is adopted here, is a perception-based approach of sound classification. In this case, the classification does not rely on mathematical descriptors anymore but rather on the results of psychoacoustic experiments. On such experiments human listeners are asked to evaluate similarity between pairs of sounds, to gather sounds in categories or to assess distance between several groups of sounds. These methods are commonly used in studies related to instrumental timbre perception [7, 8], musical emotion assessment [9] or environmental sounds categorization [10, 11]. These two complementary approaches for sound classification, i.e. descriptors-based and perception-based, aim at addressing specific questions and provide slightly different results.

1.3 *Tangisense*, a new platform for intuitive and collaborative interaction

A tangible user interface (TUI) is a user interface in which a person interacts with digital information through the physical environment. It derives from the theoretical concept of *tangible interactivity* which states that the manipulation of real physical objects is much more efficient than the utilization of virtual widgets on a screen [12]. A part of this work is based on *Tangisense*, a platform for intuitive and collaborative interaction using an interactive table coupled

[1] http://www.marsyas.info
[2] http://clam-project.org
[3] http://users.jyu.fi/ lartillo/mirtoolbox

with tangible objects [13]. This new interface relies on RFID technology (Radio-frequency IDentification), which enables the users to manipulate tangible objects equipped with one or several RFID tags. The current prototype of Tangisense is composed of an array of 1600 RFID antennas in the form of 25 squared tiles. The surface of the interactive area is about $1m^2$ divided in a 80x80 grid, enabling the detection of more than 60 tangible objects moving simultaneously on the table with a spatial precision of 1.25 cm. In terms of temporal resolution, the sampling rate of the table is 20 Hz, that is the position of a tangible object is refreshed every 50 ms.

Tangisense allows users to interact with two kinds of objects: '*tangible objects*', which can be found in the context of everyday life, physically accessible and easy to grasp by the user, and '*virtual objects*' displayed on the table using LED diodes associated to each RFID antenna. The original software architecture of Tangisense is written in the cross-platform language Java, but a bridge based on the OpenSoundControl protocol has been implemented to allow the table to communicate with a series of OSC-compatible programming environments such as Max/MSP, Pure Data, or Processing.

Since a couple of years touchable and tangible interfaces have been widely used for musical expression [14, 15, 16]. Due to its original RFID-based technology, Tangisense differs from most of these interfaces, which rely on computer vision tracking. One of the main advantages compared to camera-based tabletops is that Tangisense is not sensitive to lightning conditions, and does not require any preliminary calibration step before utilization. Thanks to the thickness of the tiles (15 cm), the interactive surface may be inserted in reduced spaces and placed in any position, for example vertically against a wall. The current version of this tangible table does not include multi-touch tactile display, but this functionality will be included in the next prototype, allowing direct interaction using fingers.

2. CREATION OF A PERCEPTUAL SONIC TEXTURES SPACE

In order to study the perception of sonic textures in a musical context, we conducted a psychoacoustic experiment where a series of participants were asked to gather examples of such textures in groups according to their perceived similarity. By summing up the results provided by all the participants and processing them through a multidimensional scaling analysis (MDS), we have been able to provide a collective map of sonic textures based on all participants' perception. This experiment was implemented in Java and used the interactive tangible table described in the previous section as control interface.

2.1 Musical sonic textures collection

For the purpose of this psychoacoustic experiment, we selected a set of 16 musical sonic textures extracted from electronic and electro-acoustic musical pieces written by various composers [4], as listed in Table 1. To be selected, musical excerpts must fulfill the main criteria defining the sonic textures, i.e. presenting both short-term random properties and constant long-term characteristics. They should also present a certain degree of abstractness, i.e. the source of the sound should not be easily identifiable, in order to make the listener focus on the acoustical attributes of the textures and avoid a categorization based on the recognition of the sound source. Each excerpt was extracted as an approximately 10 seconds duration stereophonic sound file. The experimenter adjusted the loudness accordingly.

Index	*Piece*	*Composer*
1	Bohor	Y. Xenakis
2	Concret PH	Y. Xenakis
3	Cross Over	K. Hofstetter
4	Ermine	M. Namblard
5	Ermine	M. Namblard
6	Etheraction	J-M. Couturier
7	Foley Room	A. Tobin
8	Gobi the Desert	Monolake
9	Sur la plage à l'aube	M. Redolfi
10	La légende du jaguar	M. Redolfi
11	Microsound	C. Roads
12	Microsound	C. Roads
13	Nodal	H. Vaggione
14	Orient-Occident	Y. Xenakis
15	Tune of Wind	C-W. Weng
16	De Zarb à Daf	L. Ceccarelli

Table 1. List of musical excerpts used for the sonic textures grouping experiment.

2.2 Experimental protocol

In this grouping experiment, also named *free categorization task*, subjects were asked to cluster sonic textures from a stimuli set into several groups according to their perceived similarity. As illustrated in Figure 1 and Figure 2, they performed this task by manipulating three different kinds of tangible items on the interactive table:

- 16 cubic '*sound objects*' numbered from 1 to 16, associated to one sonic texture of the stimuli set,
- 4 '*basket objects*' , creating virtual rectangular baskets on the table,
- one '*loudspeaker object*', allowing to draw a virtual circular listening area on the table

To listen to each sonic texture from the stimuli set, the participant had to place the actual tangible sound object within the listening area drawn around the tangible loudspeaker object. The volume of the sound was mapped according to the distance with the center of the listening area, and the size of the listening area let the possibility for the

[4] Material related to this study, like the collection of musical excerpts used for the listening tests and videos showing the experiment process, are available online: http://www.tele.ucl.ac.be/~jjfil/SMC10.html

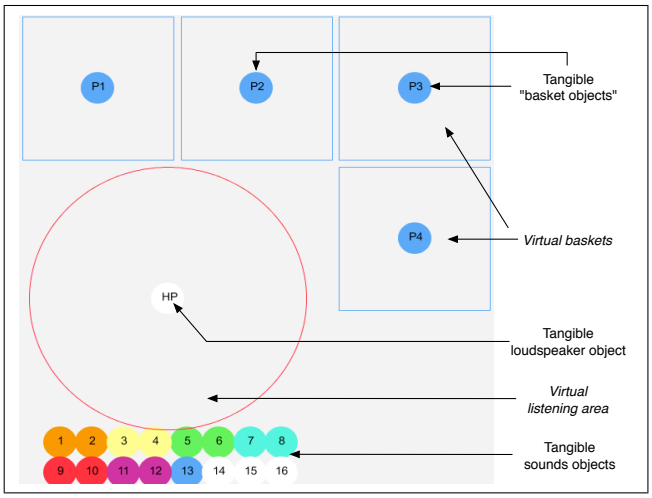


Figure 1. Representation of the tangible and *virtual* objects used in the sonic textures grouping experiment.

Figure 2. A subject completing the sonic textures grouping experiment using the tangible interactive table.

participant to listen to several sounds simultaneously. Then the participant was invited to gather the sounds in groups according to their similarity by putting the actual tangible sound objects within relevant virtual baskets. At the beginning of the experiment three baskets were available for the subject, who could ask for a fourth basket if needed. An important instruction given by the experimenter was that subject had no obligation to place each sonic texture in a group, in order to avoid a 'garbage basket' including heterogeneous sounds. At the end of the classification, the subject could thus leave one or several sounds unclassified, if he or she thought these sounds did not share enough similar characteristics to belong to any group.

In order to make the participant feel comfortable with both the categorization task and the manipulation of the tangible interface, time was not limited and even not measured as an indicator of subject's ability. The average duration for each user to achieve the sounds gathering was approximately fifteen minutes.

Once grouping completed, each participant answered an oral questionnaire where they were first asked to evaluate both the difficulty of the sound grouping task and the utilization of the tangible interface to complete this task. Participants were then asked to explain their categorization by describing the method they adopted to gather the sounds, and by specifying the content of the baskets by associating keywords to each class of sonic textures.

2.3 MDS analysis

For each subject participating to this experiment, a sonic textures distance matrix was created based on the result the sounds gathering. The creation of this distance matrix relied on a straightforward Boolean rule: if sounds belonged to the same group, their mutual distance was set to zero, in this opposite case, their distance was set to 1. As a consequence, leftover sounds had a distance of 0 between themselves and 1 with any other sounds. Figure 3 shows an example of sonic textures classification and the resulting individual, symmetrical and binary distance matrix.

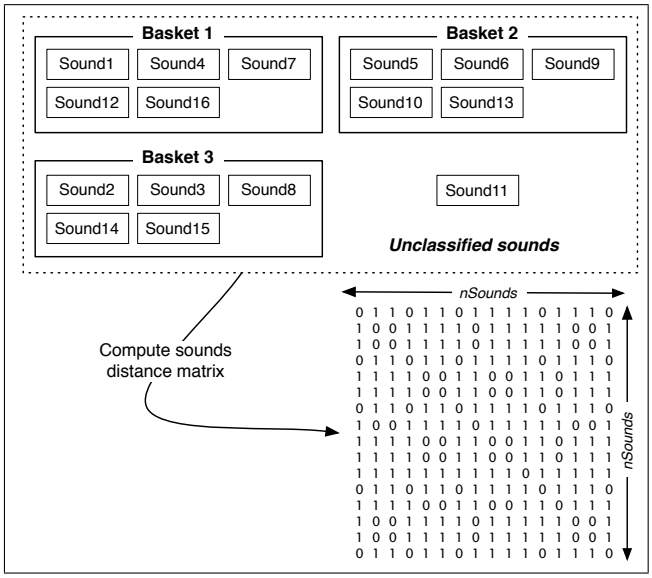


Figure 3. At the top, an example of sonic textures grouping, at the bottom the resulting distance matrix.

23 participants, aged from 13 to 60 and having various degrees of musical training and experience in sound processing, performed the experiment. A global distance matrix was then put together summing all the 23 individual binary matrices. With this collective matrix, we applied a multidimensional scaling analysis (MDS) in order to construct a best-fitting geometric map of the sonic textures stimuli set.

Multidimensional scaling is a set of statistical techniques widely used in information visualization to explore similarities or dissimilarities in data [17]. It takes as an input a matrix of inter-point distances, summarizing the dissimilarities measured in the input data, and creates a configuration of points. An iterative algorithm is used to optimize the relative position of the data on the map to match the dissimilarities, which can be issued from different types of measurements. Ideally, the resulting space is in two or three dimensions, and the Euclidean distances between

them reproduce the original distance matrix. This method has been widely used in earlier works related to perceptual sound classification, especially for the creation of perceptual representation of instrumental timbre [7, 8].

Figure 4 shows the 2-D and 3-D perceptual spaces of sonic textures resulting from the MDS analysis results of the 23 participants provided after the grouping experiment.

2.4 MDS evaluation

For the purpose of our study, we fed the MDS algorithm with the 16-by-16 collective sonic textures distance matrix D, corresponding to the sum of all the individual distance matrices. MDS returned an 16-by-16 configuration matrix Y, whose rows were the coordinates of the 16 reconstructed points in a 16-dimensional space; this space could be reduced by considering only the k first columns of Y, corresponding to the reconstruction of the 16 points in a *k-dimensions* space. In our case, the evaluation of the MDS analysis mainly consisted of determining if the reduction to a 2-D or 3-D space, easy to visualize, provided a reasonable approximation to the original distance matrix D.

A common method consists of evaluating the sorted eigenvalues of the scalar product matrix Y*Y': the relative magnitudes of those eigenvalues indicate the relative contribution of the corresponding columns of Y in reproducing the original distance matrix D with the reconstructed points. If only the first two or three eigenvalues are large, then only those coordinates of the points in Y are needed to accurately reproduce D. Figure 5 displays a scree plot of the scalar product matrix Y*Y' resulting of our MDS analysis, i.e. the eigenvalues of this matrix, normalized and sorted in a descending order of magnitude. We can see two large positive eigenvalues on the scree plot, which means that the configuration of points created by the MDS can be plotted in a 2D map. The four negative eigenvalues indicate that the distances on the reconstructed space are not Euclidean, that is no configuration of points can exactly reproduce the original distance matrix D issued from the sounds grouping experiment. Fortunately, these negative eigenvalues are relatively small in comparison to the largest positive ones, which indicate that the reduction to the two first columns of Y provide a fairly accurate approximation of the distance matrix D. For a detailed description of the multidimensional scaling analysis and its evaluation methods, interested reader can refer to [17].

2.5 Discussion

Over the 23 participants who performed this experiment, 65% classified the sounds set in three groups and 35% in four groups. More than 60% of the baskets contained 4 or 5 sounds in the case of the 3-groups classification and 3 or 4 sounds in the case of the 4-groups classification. The average number of isolated sounds, i.e. not gathered with any other sounds, was 1.2 in the case of the 3-groups classification and 1.375 in the case of the 4-groups classification. Eight participants did not leave out any sound.

During the step of the verbal explanation, where subjects were asked to describe the groups according to one or several keywords, interesting information came up regarding the way the sonic textures were perceived by the participants and which were criteria used by the participants to make their own classification. Four main categories came out among the keywords provided by the subjects:

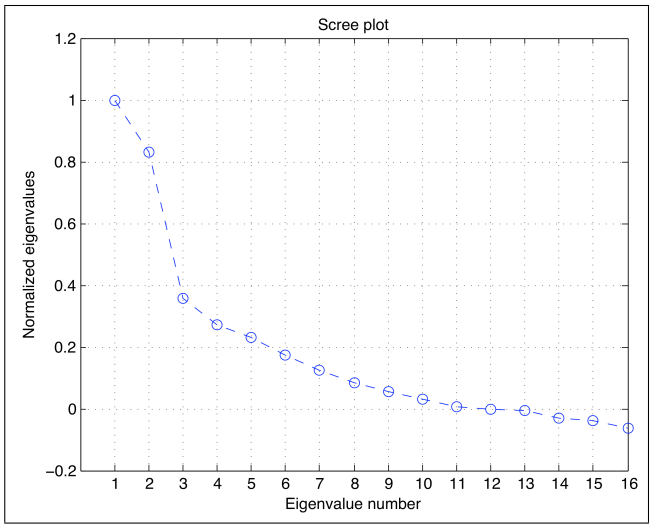


Figure 5. Scree plot of the scalar product matrix Y*Y, showing the sorted eigenvalues, normalized by the maximal eigenvalue, as a function of the eigenvalue index.

- keywords describing the sound source evoked by the sonic texture: *water*, *nature*, *machines*, *factories*, *etc...*
- keywords describing the emotion evoked by the sonic texture: *relaxing, unpleasant, frightening, annoying, etc...*
- keywords related to the form of the texture, somehow linked to its temporal profile: *continuous sounds, fluid sounds, grainy sounds, smooth sounds, rhythmic sounds, percussive sounds etc...*
- keywords related to the color of the texture, which may somehow related to its spectral content: *bright, metallic, etc...*

It was interesting to note that some subjects could use the same keywords to characterize groups that included different sounds.

In terms of interaction, people agreed about the value of the tangible interface for such a free categorization task, which may sometimes turn out to be unpleasant when using a traditional mouse-based user interface. Benefits offered by such interfaces for organizing, exploring and classifying data have already been proved by the computer-human interaction community [18]. In our setting, the manipulation of physical tangible objects was quick and intuitive, and the possibility offered to the user to listen to several sounds simultaneously was very helpful to validate the grouping.

Primary conclusions can be drawn from the observation of the perceptual sonic textures spaces resulting of this psychoacoustic experiment (Figure 4). First of all, it is worth

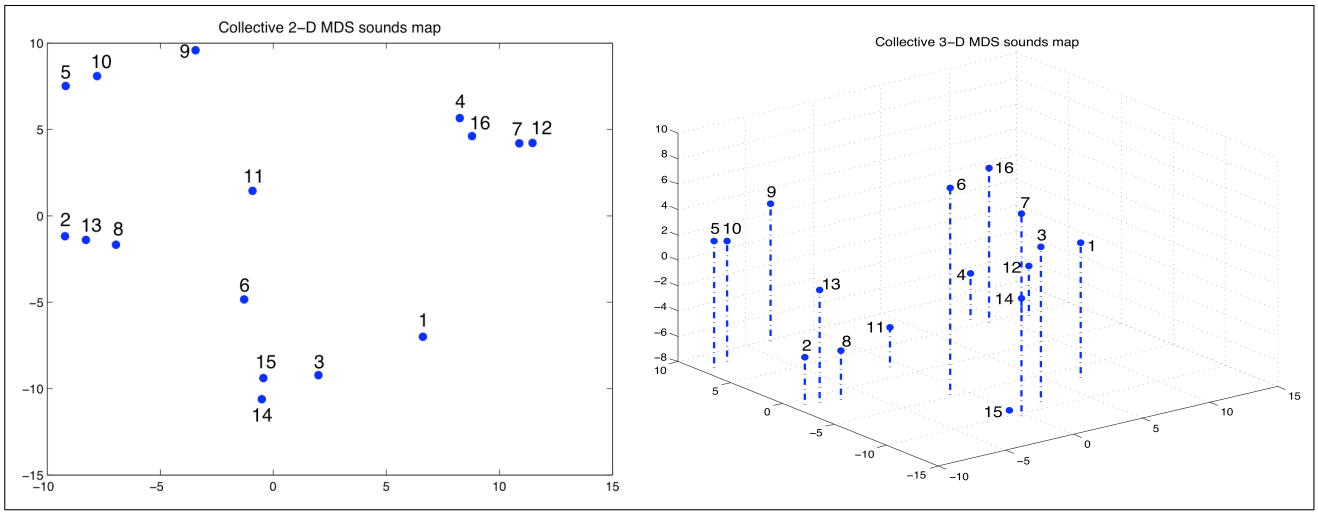


Figure 4. 2-D and 3-D sonic textures spaces resulting of the MDS analysis.

mentioning that the positions of the sounds were relatively scattered all around the space, which reflects some kind of variability between the subjects' answers. However, clusters of sounds clearly appear in the sound space, which indicate certain coherence between the subjects' answers. For instance, 83% of the participants agreed to gather textures 5 and 10 in the same basket. These two observations - variability and consistency of subjects answers - tend to indicate that the grouping of these textures was not a trivial task.

2.6 Future tracks

This paper describes an ongoing long-term research work focused on the analysis and synthesis of sonic textures. The next step of this research will aim at attempting to interpret the geometrical configuration of the sonic textures in the perceptual map issued from the MDS analysis with regard to the factors that may explain the ordering of the points along the various dimensions of the space. We are going to describe specifically each sonic texture in terms of a series of signal descriptors typically used in music information retrieval [6], including spectral descriptors (spectral centroid, spectral spread), temporal descriptors (RMS energy, zerocrossing rate) and spectro-temporal descriptors (spectral flux, roughness). Once those features will have been combined, we will try to determine if one or several of these signal descriptors can be mapped to the dimensions of the perceptual space.

One of our assumptions, inspired by Schaeffer's approach on morphological description of sound [19], is that perception of sonic textures could be based on two main attributes, i.e. the color and the form of the texture. We assume that color of a sonic texture is related to its spectral content and could be somehow characterized by its spectral features. By form, we would like to address the notion of granularity of the textures. A descriptor based on a short-term analysis of the temporal envelope evolution could enable to quantitatively estimate this attribute. Recent works proposed by [20] and introducing a hybrid model of sound relying on both concepts of sound color and density could be an inspirational track for the following of our research.

Among the other tracks we like to address in a near future, we plan to extend the MDS analysis by treating the collective distance matrix issued form this sound grouping experiment with a hierarchical clustering algorithm. Hierarchical clustering is a way to investigate grouping in data over a variety of scales, by creating a cluster tree [21]. This could provide complementary information to the MDS analysis. Another track we would like to investigate is the creation of an individual sonic textures space, relying on results of psychoacoustic experiments of a single subject. For that, we plan to conduct another series of listening tests including the grouping experiment followed by a second dissimilarity experiment. In this second experiment, participants will be asked to refine the classification issued from the grouping experiment by numerically rating the distance between sonic textures belonging to the same group. This would allow the creation of an individual sounds distance matrix which could then be analyzed by multidimensional scaling. This experiment has already been prototyped with the tangible interface and should be conducted in the following weeks.

3. INTERACTIVE AND TANGIBLE EXPLORATION OF A SONIC TEXTURES SPACE

This second part of our research aimed at investigating how a tangible interactive table such as Tangisense could be used for a gestural interaction with sonic textures in real-time. In order to achieve this, we designed a musical tabletop application inspired by the metaphor of the exploration of a gravitational sonic textures space.

3.1 Related works

Recently, there has been a growing interest towards experimenting multi-touch and tangible devices for musical purposes. From pioneer works, such as *Audiopad* [22] or the *Reactable* [14], to the most recent ones like the *Bricktable* [16], artists and researchers have highlighted the possibilities offered by such interfaces for musical expression

and performance. Some of these studies were dealing with the use of a tabletop interface for the exploration of sonic spaces, but addressed this question according to a MIR-oriented approach aiming at facilitating the navigation in large music collections [23, 24]. This is a slightly different approach than the one adopted in this present study, described in the next section.

3.2 A gravitational sonic space

As shown in Figure 6, we considered the sonic space as a gravitational system populated by n '*sounds attractors*' which exert attraction force on m '*sound actuators*' depending on their relative distance. Each attractor has its own mass, represented by the radius of its surrounding circle, which is proportional to the attraction force it exerts. The sonic interaction is designed as follow: a sound, taken from the set of sonic textures used in the psychoacoustic experiment presented above, is associated to each attractor. Each actuator generates an audio track composed of the mix of the n sonic textures associated to the attractors that are present in the space. The weight of each sonic texture in the mix depends on the forces exerted by the corresponding attractors on the actuator. In order to enhance the possibilities of interaction, we added to the actuators the ability to process the sonic textures through live granulation.

This musical experiment was developed under Max-MSP, which communicated with the interactive table through OSC. The exploration of the sonic space partly relied on interpolation tools developed by [25] and on GMU, a flexible granular synthesis environment in Max/MSP [26].

3.3 Interaction with the tangible table

As shown in Figure 7 and Figure 8, the exploration of the sonic textures space is performed by manipulating three kinds of tangible objects [5] :

- 6 hemispheric tangible '*sound actuators*',
- 16 cubic tangible '*sound attractors*',
- one '*tangible rubber*',

The tangible sound actuators are used to physically navigate in the sonic space. These objects are hemispheric and equipped with five RFID tags; beside increasing the robustness of the detection, it allows to provide information related to the inclination of the object on the table. We also chose to map the sound granulation parameters, i.e. grains length and triggering, to the rotation of the actuators.

The initial idea was to limit the exploration to a fixed perceptual sonic textures space, i.e. resulting of the psychoacoustic experiment described in section 2. However in order to increase the interactivity and enhance the possibilities of expression, we chose to give the performer the ability to freely design, populate and modify the sonic space by handling the tangible objects himself. At the beginning

[5] a video demonstrating this musical tabletop application is available online: http://www.tele.ucl.ac.be/~jjfil/SMC10.html

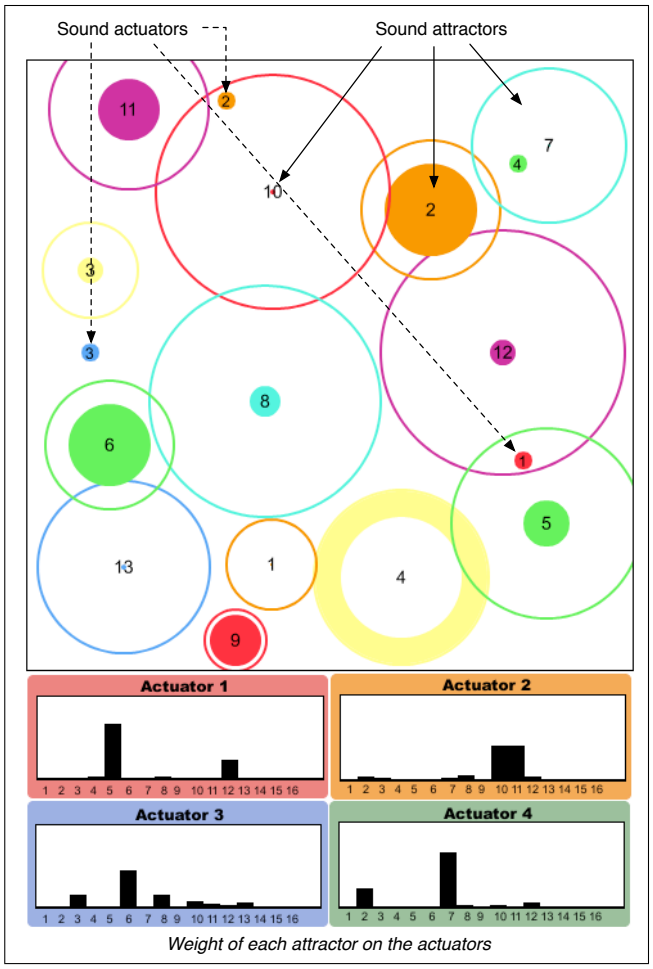


Figure 6. Representation of a sonic textures space populated by 13 sound attractors of various mass and 4 sound actuators.

of the performance, the user could choose to load the configuration of the attractors corresponding to the perceptual map issued from the psychoacoustic experiment, or to start from an empty space and populate the sonic space by placing tangible attractors on the table. Sound attractors are represented by illuminated areas on the table; to prevent a cluttering on the table and facilitate the navigation into the sonic space, attractors were persistent, which means that they would keep their position in the sonic space even if the corresponding tangible sound object was removed from the table. A tangible rubber is placed at the disposal of the performer to remove any unwanted sound attractor.

3.4 Evaluation and future tracks

This musical tabletop application constitutes the first attempt to control real-time sound processes using the RFID-based tangible interface. The mapping between gestures and sound parameters designed in this application allows the performer to interact with sonic textures through a large range of musical gestures: while a seamless navigation in the sonic space is made possible by sliding the tangible sound actuators along the table, drawing actuator trajectories through rapid gestures allows to shape the sound matter intuitively. Multilayered soundscapes can be composed

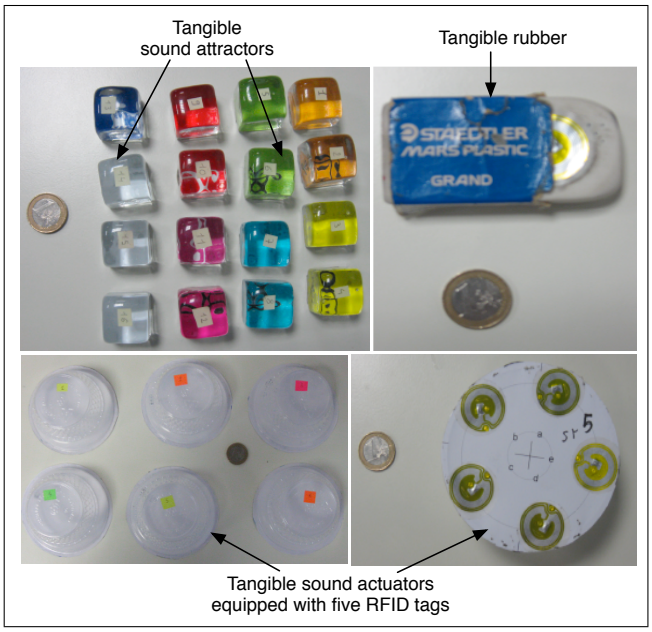


Figure 7. The three kinds of tangible objects used for exploring the sonic textures space.

by fixing the position of some actuators in specific areas of the sonic space while moving around other actuators in different regions of the table. The exploration of the sonic space by several performers is also possible (Figure 9) and demonstrates the potential of such kind of interfaces for collaborative musical improvisation. Although the sonic space browsed in the first version of this instrument was composed of sonic textures selected for the purpose of our psychoacoustic experiment, it can easily be extended to any sound source. Such interface providing intuitive interaction with sound could thus be of interest for musicians or phonographers using field recordings as a primary sound material in their compositions.

4. CONCLUSION

The study described here above included two main parts: the first one aimed at building a space of sonic textures based on human perception. For that purpose, we conducted a psychoacoustic experiment where 23 participants were asked to gather a set of musical sonic textures extracted from various electronic or electro-acoustic pieces in a few groups according to their perceived similarity. This experiment relied on a new interactive and tangible interface based on RFID technology. A multidimensional scaling analysis (MDS) was applied to the results provided by the 23 subjects who completed the sounds grouping experiment, enabling to build both 2-D and 3-D perceptual sonic textures spaces. An important track we would like to investigate in a near future is the likely correspondence between audio descriptors of the sonic textures and dimensions of a perceptual sound space. The second part of this study was performance-oriented and aimed at investigating the possibilities offered by a tangible interface for the gestural control of sonic textures. We also designed a musical tabletop application enabling to navigate in a gravitational

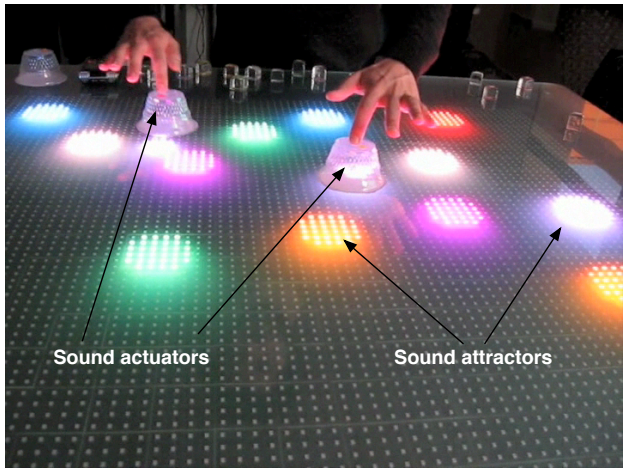


Figure 8. Interactive exploration of the sonic textures space using tangible sound actuators and attractors.

Figure 9. Collaborative exploration of the sonic space by two performers

sonic texture environment through a tangible interaction. The first attempts to test this application brought up some very promising insights regarding the possibilities offered by such interfaces for the real-time processing of sonic textures.

5. ACKNOWLEDGMENTS

Authors would like to thanks the 23 subjects who participated in the psychoacoustic experiment described in this paper. We are also very grateful to Valentin Valls for his precious help in programming interactions on the tangible table. This work was partly supported by the COST IC 0601 Action on Sonic Interaction Design [6] (SID) and Numediart [7] a long-term research program centered on Digital Media Arts and funded by the Région Wallonne, Belgium.

[6] http://www.cost-sid.org
[7] http://www.numediart.org

6. REFERENCES

[1] P. Kokoras, "Towards a holophonic musical texture," *The Journal of Music and Meaning*, vol. 4, 2007.

[2] J.-J. Filatriau and D. Arfib, "Instrumental gestures and sonic textures," in *Proceedings of the Sound and Music Computing Conference (SMC'05)*, Salerno, Italy, 2005.

[3] G. Strobl, G. Eckel, and D. Rocchesso, "Sound texture modeling : A survey," in *Proceedings of the Sound and Music Computing Conference (SMC'06)*, Marseille, France, 2006.

[4] J.-J. Filatriau, D. Arfib, and J.-M. Couturier, "Using visual textures for sonic textures production and control," in *Proceedings of the International Conference on Digital Audio Effects (DAFx'06)*, Montreal, Canada, 2006.

[5] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, 2002.

[6] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," tech. rep., Ircam - Centre Georges Pompidou, 2004.

[7] J. M. Grey, "Multidimensional perceptual scaling of musical timbre," *Journal of the Acoustical Society of America*, vol. 61, pp. 1270–1277, May 1977.

[8] S. McAdams, S. Winsberg, S. Donnadieu, G. D. Soete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: common dimensions specificities, and latent subject classes," *Psychological Research*, vol. 58, pp. 177–192, 1995.

[9] S. Vieillard, E. Bigand, F. Madurell, and J. Mazoreau, "The temporal processing of musical emotion in a free categorization," in *Proceedings of the 5th Triennal ESCOM Conference, Hanover, Germany*, 2003.

[10] N. Saint-Arnaud, "Classification of sound textures," Master's thesis, Massachussets Institute of Technology, 1995.

[11] O. Houix, G. Lemaître, N. Misdariis, P. Susini, K. Franinovic, J. Otten, J. Scott, Y. Visell, D. Delvallez, F. Fontana, S. Papetti, P. Polotti, and D. Rochesso, "Everyday sound classification: sound perception, interaction and synthesis, part 1: state of the art," tech. rep., CLOSED project deliverable 4.1, 2007.

[12] E. Hornecker and J. Burr, "Getting a grip on tangible interaction: a framework on physical space and social interaction," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, Montréal, Canada, 2006.

[13] S. Kubicki, S. Lepreux, Y. Lebrun, P. D. Santos, C. Kolski, and J. Caelen, "New human-computer interactions using tangible objects: application on a digital tabletop with rfid technology," in *Proceedings of the 13th International Conference on Human-Computer Interaction (HCII'09)*, San Diego, USA, 2009.

[14] S. Jordà, M. Kaltenbrunner, G. Geiger, and R. Bencina, "The reactable," in *Proceedings of the International Computer Music Conference (ICMC'05), Barcelona, Spain*, pp. 579–582, 2005.

[15] G. Levin, "The table is the score: an augmented-reality interface for real-time, tangible, spectrographic performance," in *Proceedings of the International Computer Music Conference (ICMC'06), New Orleans, USA*, 2006.

[16] J. Hochenbaum and O. Vallis, "Bricktable: A musical tangible multi-touch interface," in *Proceedings of Berlin Open Conference '09. Berlin, Germany*, 2009.

[17] L. Borg and P. Groenen, *Modern multidimensional scaling: theury and applications*. Springer Series in Statistics, 2005.

[18] R. Jabob, H. Ishii, G. Pangaro, and J. Patten, "A tangible interface for organizing information using a grid," in *Proceedings of the SIGCHI conference on Human factors in computing systems (SIGCHI'02)*, Minneapolis, USA, 2002.

[19] P. Schaeffer, *Traité des Objets Musicaux*. Seuil, 1966.

[20] H. Terasawa, *A hybrid model for timbre perception: qualitative representations of sound color and density*. PhD thesis, Stanford University, 2009.

[21] S. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, pp. 241–254, September 1967.

[22] J. Patten, B. Recht, and H. Ishii, "Audiopad: a tag-based interface for musical performance," in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME'02)*, Dublin, Ireland, 2002.

[23] D. Diakopoulos, O. Vallis, J. Hochenbaum, J. Murphy, and A. Kapur, "21st century electronica: Mir techniques for classification and performance," in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, 2009.

[24] C. Julia and S. Jordà, "Songexplorer: a tabletop application for exploring large collections of songs," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan, 2009.

[25] T. Todoroff and L. Reboursière, "1-d, 2-d and 3-d interpolation tools for max/msp/jitter," in *Proceedings of the 6th Sound and Music Computing Conference (SMC'09)*, Porto, Portugal, 2009.

[26] C. Bascou and L. Pottier, "Gmu, a flexible granular synthesis environment in max/msp," in *Proceedings of the Sound and Music Computing Conference (SMC'05)*, Salerno, Italy, 2005.

AV CLASH – ONLINE TOOL FOR MIXING AND VISUALIZING AUDIO RETRIEVED FROM *FREESOUND.ORG* DATABASE

Nuno N. Correia
Aalto University, School of Art and Design
Media Lab
mail@nunocorreia.com

ABSTRACT

In this paper, the project *AV Clash* will be presented. *AV Clash* is a Web-based tool for integrated audiovisual expression, created by Video Jack (the author and André Carrilho, with the assistance of Gokce Taskan). In *AV Clash*, users can manipulate seven "objects" that represent sounds, incorporating audio-reactive animations and graphical user interface elements to control animation and sound. The sounds are retrieved from online sound database *Freesound.org*, while the animations are internal to the project. *AV Clash* addresses the following research question: how to create a tool for integrated audiovisual expression, with customizable content, which is flexible, playful to use and engaging to observe? After an introduction to the project, a contextualization with similar works is presented, followed by a presentation of the motivations behind the project, and past work by Video Jack. Then the project and its functionalities are described. Finally, conclusions are presented, assessing the achievement of the initial aims, and addressing the limitations of the project, while outlining paths for future developments.

1. INTRODUCTION

AV Clash is a Web-based project by Video Jack (the author and André Carrilho, with the assistance of Gokce Taskan), which allows for the creation of audiovisual compositions, consisting of combinations of sound and animation loops. *AV Clash* is composed of seven audiovisual units, which enable playback and manipulation of four different loops of sound and audio-reactive visuals (one combination of audio and visuals at a time). These units were named "Interactive AudioVisual Objects" ("IAVOs") since they are composed of user interface (UI) elements that trigger and manipulate sounds, together with animations that react to those sounds. The sounds in *AV Clash* are retrieved from *Freesound.org*, an online sound database. The animations were developed by André Carrilho.

AV Clash is still being developed, and tested, by Video Jack. Therefore it is not online yet, although Video Jack have already registered the domain www.avclash.com, where the project will be hosted. For now, this domain points to a page where demonstration tracks recorded with *AV Clash* can be listened to. No public presentations (performances or exhibitions) have been made with the project yet.

2. CONTEXTUALIZATION

AV Clash follows a long tradition of explorations towards integration of sound and image. Ancient Greek philosophers, such as Aristotle and Pythagoras, considered that there was a correlation between the musical scale and the rainbow spectrum of hues [4]. The color to music correlation was further explored in the Renaissance by several artists, including Leonardo da Vinci, and later by Isaac Newton [9, pp. 45-46]. Newton's experiments influenced the creation of the *Ocular Harpsichord*, an early "color organ" by Father Louis Bertrand Castel, around 1730 [4].

Wallace Rimington created his electric *Colour Organ* in 1893, which "mixed primary colors into more nuanced hues that could be projected on gently-moving gauzy curtains to obtain polymorphous color flows" [3]. This *Colour Organ* inspired composer Alexander Scriabin to write "a scenario of changing colors into the score of his 1910 Prometheus symphony" [3]. The tradition of color organs continued into the mid 20th century, and influenced abstract filmmakers – "for the cinema, with its standardized methods of production, reproduction and exhibition, seemed the ideal vehicle for Color Music" [3].

In the 1920s, Oskar Fischinger and Walther Ruttman created "visual music" films in Germany – a combination of tinted animation with live music [4]. Oskar Fischinger moved to Hollywood in 1936, becoming an inspiration to a younger generation of visual music artists, such as Jordan Belson, Harry Smith and brothers John and James Whitney. The Whitney brothers "decided to take up abstract animation after seeing a screening of Oskar's films" [3]. John Whitney is "widely considered 'the father of computer graphics'" for his explorations of computer-generated visuals through mathematical functions [6, p. 15]. He was among the first generation to use computers for the creation of artworks in the 1960s.

Progress in computing hardware played an important role in the dissemination of digital art from the late 20th century onwards. Sound is one of the major areas of exploration for digital artists. Artistic digital sound and music is a vast territory, that includes: "pure sound art (without any visual component), audio-visual installation environment and software, Internet-based projects that allow for real-time, multi-user compositions and remixes, as well as networked projects that involve public places or nomadic devices" [6, p. 133].

These digital sound and music projects are frequently interactive, and some of them incorporate visuals: "(they) also commonly take the form of interactive installations or 'sculptures' that respond to different kinds of user input or translate data into sounds and visuals" [6, p. 136]. Many of these projects that combine music and visuals digitally "stand in the tradition of kinetic light performance or the visual music of the German abstractor and painter Oskar Fischinger" [6, p. 134]. Among the artists that explore integrated audiovisual expression by digital means are: Golan Levin, notably with his *Audiovisual Environment Suite*; Toshio Iwai, with projects such as his recent *Electroplankton* and *Tenori-On*; and John Klima, namely with *Glasbead*, an "online art work that enables up to 20 simultaneous participants to make music collaboratively via a colorful three-dimensional interface" [8, p. 54].

Internet proved to be a fertile territory for developing digital sound and music projects, exploring the possibilities of connecting different musicians, sound artists, and their audiences. In 1998, Sergi Jordà created the first version of *FMOL*, "an Internet-based music composition system that could allow cybercomposers to participate in the creation of the music for La Fura's next show, F@ust 3.0 (…) freely inspired by Goethe's work" [2, p. 326]. Like *Glasbead*, it allowed for online collaborative music composition.

Freesound Radio [1] (2009) is another example of Web-based sonic collaboration. It is an online "experimental environment that allows users to collectively explore the content in *Freesound.org* by listening to combinations of sounds represented using a graph data structure" [7, p. 1]. *Freesound Radio* retrieves sounds from *Freesound.org*, "one of the most widely used sites for sharing sound files licensed under a Creative Commons (CC) license" [7, p. 1].

3. MOTIVATION AND PREVIOUS WORK

AV Clash is Video Jack's fourth major interactive audiovisual project, after *Heat Seeker* (2006), *AVOL* (2007) and *Master and Margarita* (2009).

Among previous Video Jack projects, the most direct predecessor of *AV Clash* is *AVOL* [2]. *AVOL* allows for the integrated manipulation of sound and visual elements. The project is composed of seven "objects", which enable triggering four possible combinations of sound and visuals. These "objects" integrate graphical user interface elements to manipulate the audiovisual combinations. Objects can be moved around the screen, and object collisions generate special animations and sounds.

After the conclusion of *AVOL* in 2007, and its presentation in several festivals in 2008, the author detected several limitations in the project. Among these limitations are: a fixed number of sound and animation loops; a small degree of audio and visual manipulation possibilities; difficulty in making simultaneous changes in multiple objects; absence of recording or sharing capabilities; difficulties in identifying each object; and lack of collaboration functionalities.

Video Jack started developing a new project in early 2010, entitled *AV Clash*, to address the limitations of *AVOL*. In order to allow for a greater audio flexibility, the author decided to connect this new project to an online sound database. *Freesound.org*, with its vast repository of *Creative Commons* licensed sounds and its tag-based audio categorization, seemed to be adequate. The precedent of *Freesound Radio*, which is successful in retrieving sounds from *Freesound.org* for sonic composition, also pointed out in this direction. The animations would still be developed by Video Jack, but it would be possible for users to choose among a list of different animations associated with a certain tag. The author invited a former student, Gokce Taskan, to collaborate on the programming side of the development. Because of the importance of vector animation for the project, *Freesound Radio*'s successful usage of *Flash*, and the experience gained with previous projects, Video Jack decided to continue using Adobe *Flash* for the development of *AV Clash*.

AV Clash expands on *AVOL*, by addressing the following research question: how to create a tool for integrated audiovisual expression, with customizable content, which is flexible, playful to use and engaging to observe?

4. DESCRIPTION OF THE TOOL

4.1 Image and Sound Association

Each IAVO has a "tag" associated to it, which is retrieved from *Freesound.org*. In a first stage of development, 10 *Freesound.org* tags will be supported, chosen from its most popular tags. *AV* Clash contains seven animations per tag, in a total of 70 animations. The tag of each IAVO acts as a filter to the possible four sounds and four animations it may contain. Each tag is associated with a color. During one *AV Clash* session, the user may change sounds and animations, and even the tag, of each IAVO. More tags and animations will be added in later stages of development.

4.2 Start Screen and Stage

When users enter *AV Clash*, they are presented with a screen composed of seven colored vertical bars, of equal width – the stage. Each bar, colored according to the associated tag, represents a different activation point for

[1] http://radio.freesound.org/

[2] http://www.videojackstudios.com/projects/avol

each IAVO. When the project starts, the sounds are loaded from online sound database *Freesound.org*. As the 28 sounds are loading, a circular pre-loader appears near the bottom of the screen, to represent the loading process of the four sounds associated with each IAVO. The name of the tag is shown above the circle. A percentage is shown in the center of the circle, displaying the total loaded percentage of the four sounds.

The selection of initial tags, sounds and animations is random. Seven random tags are selected by the software, and then four random sounds and animations are selected per tag, among the ones associated with that tag. The sounds are not picked among the totality of sounds available per tag, but rather from the 20 most popular sounds with that tag (based on number of downloads from *Freesound.org*).

After all 28 sounds have been loaded, the bars disappear – they split up in two at the point where the pre-loader was, and retracts into the top and bottom edges. The IAVOs appear in the place of the pre-loaders, but they are not playing yet.

The stage is resizable, adapting to changes in the browser size. However, there is a minimum size, beyond which it does not shrink further (800 by 600 pixels).

4.3 "Stopped IAVO" User Interface and Functionalities

When an IAVO is not playing, it shows a limited set of options. If the user is not rolling over the object with the cursor, only an outer "ring" is shown, and a central button, colored according to the object's tag. This ring allows for the IAVO to be dragged and "thrown" on the stage. When the cursor rolls over the object, additional options are shown: four audiovisual loop selection buttons, and a red button, which deletes the object (Figure 1.). The tag of the object is also shown above the IAVO.

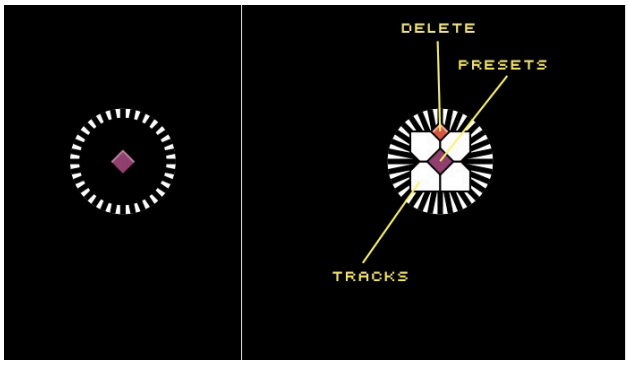


Figure 1. Stopped IAVO, in rollout (left) and rollover (right) states

4.4 "Playing IAVO" User Interface and Functionalities

The UI of each playing IAVO is composed of nine buttons and two sliders (Figure 2.).

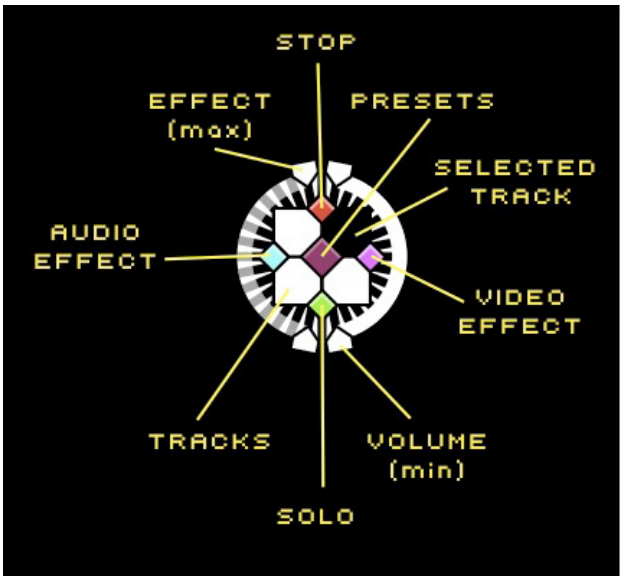


Figure 2. UI of playing IAVO, with second sound selected

4.4.1 Presentation of Main UI Elements

The UI is arranged around and inside the IAVO's "ring", which allows for dragging and throwing the object around the stage. The ring also incorporates two faders, shaped as semi circles on each of the sides. Inside of the ring are nine buttons, four of which arranged in a two-by-two grid (the "selection buttons"), with four additional buttons in the outside middle points of the grid (red stop button on top, green "solo" button on bottom, cyan audio effect button on the left, and magenta visual effect button on the right), and one more button in its center (the "back" button).

Different UI elements are visible, depending on user interaction and the position of the cursor relatively to the IAVO. In its "passive" state, when the user is not interacting with it, only the outer half of the ring and part of the faders are shown – the fader thumbs are hidden (Figure 3., left). When the user rolls over the outer half of the ring, its inner half appears, and also the fader thumbs (Figure 3., right), allowing for the manipulation of the faders. Rolling over within the inner half of the ring causes the whole UI of the IAVO to appear (Figure 2.), and also the tag of the object, shown above the IAVO.

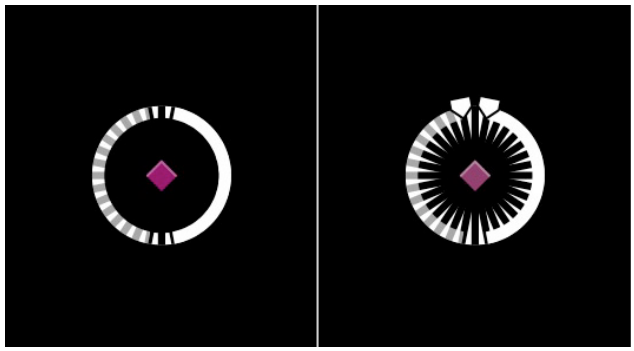


Figure 3. UI of playing IAVO in "roll out" state (left) and outer ring "roll over" (right)

4.4.2 Buttons and Faders

Pressing one of the selection buttons causes a switch in the audio and animation loops currently playing. The button relative to the audiovisual loop playing is always hidden.

Pressing the red button stops the sound and animation currently playing (the IAVO switches to "stopped" position). The green button "solos" the IAVO, stopping all the remaining ones. When an IAVO is stopped, all its settings (volume, effect selection and setting) are reset.

The center button (colored according to the tag assigned to the IAVO) reveals additional audio and animation selection and manipulation options (the "back" of the object, described below).

Pressing the cyan button modifies the four loop selection buttons, coloring them in cyan. The selection buttons become audio effect selection buttons. There are four audio effects per IAVO: filter, phaser, distortion and delay. The button relative to the selected effect is hidden. Selecting an effect or pressing the cyan button again reverts the selection buttons to white (they become loop selection buttons again). The filter effect is activated by default (Figure 4.).

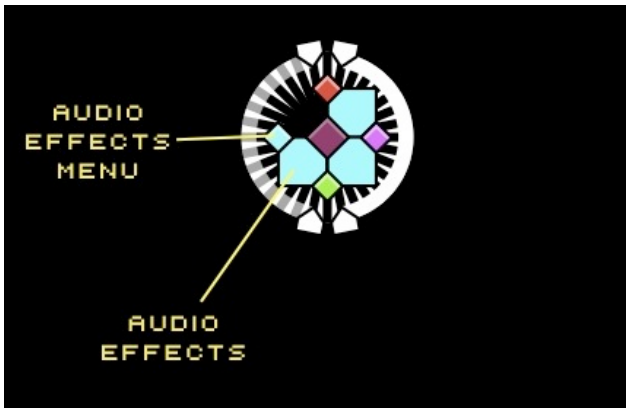


Figure 4. UI of playing IAVO with audio effects selection buttons (cyan) activated

The magenta button behaves similarly to the cyan button. The selection buttons become magenta, allowing for selection of visual effect. Visual effects determine how the animation of the IAVO reacts to its audio loop. Each animation consists of two parts – one that is audio reactive, and another one that is not. The non-reactive element serves the purpose of making the IAVO always visible, even when the visual effect makes its audio reactive component occasionally disappear. The default visual effect is scale: the size of the animation decreases or increases proportionally to the amplitude of the sound. The remaining behaviors are: opacity (opacity of animations react to sound amplitude); blur (blur level of animations is inversely proportional to the sound amplitude); and RGB transformation (red, green and blue values of the animation are transformed proportionally to sound amplitude).

Independently of the color of the selection buttons, rolling out of the IAVO and rolling in again presents the loop selection buttons (and not the cyan or magenta effect buttons).

The left fader controls the intensity of the audio effect. Its default value is zero (minimum). The right fader controls the volume of the sound, and the size of the animation (both its reactive and non reactive elements). Its default position is in the middle.

4.5 Dragging, Throwing and Clashes

An IAVO can be dragged and repositioned on the stage. It can also be "thrown", by dragging and releasing the IAVO in motion. The "throw" behavior causes the IAVO to continue moving in the direction and the speed it had when released, indefinitely, until the user clicks on it again, or presses the "back stage" button. If the IAVO reaches the edges of the screen, it starts moving in the opposite direction, with the same speed.

When a moving IAVO hits another object, a clash animation occurs, and a clash sound is triggered within the static object. The IAVO starts moving in the opposite direction. The collision sound consists of a one second random segment, from one of the four sounds of the static IAVO that is currently not playing (picked randomly), with a delay effect applied to it.

4.6 IAVO User Interface and Functionalities – "Back"

When the user presses the center button of the IAVO, an animation occurs, transforming the UI of the object: the four selection buttons expand into four larger rectangles, and the central button is enlarged and rotated. All the remaining previous UI elements disappear, with the exception of the ring, which still is partially visible underneath the new four rectangles (Figure 5.).

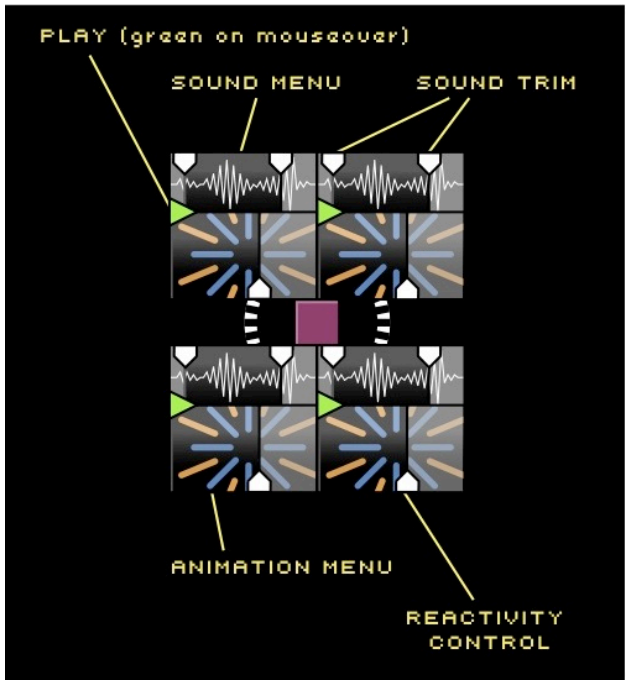


Figure 5. Back of the IAVO, in rollover state

Each of the four new rectangles allows users to pick and adjust sounds and animations for the respective four loop selection buttons of the IAVO. Each rectangle is composed of two elements: a graphical representation of the sound loop, on the top, and an image from the animation, on the bottom. Sliders on top allow for adjusting the start and stop positions for the loop. A slider on the bottom allows for adjusting the reactivity of the animation to the sounds, since some sounds might have a lower dynamic range than others. Moving this "audio reactive sensitivity" slider to the right compensates for a lower sound dynamics. The default position of this slider is center.

By clicking in the sound image, a pop-up menu appears, allowing for the selection of other sounds from *Freesound.org* with the same tag. The pop-up menu includes the following information per sound: file name; short description; duration; author name. The pop-up appears up or down from the point where the user has clicked, depending where there is more space in the screen. When one sound is selected, a pre-loader animation is triggered – a circle starts to be drawn around the ring of the IAVO. When the circle is fully drawn, the sound has finished loading.

Clicking in the animation image activates a pop-up with images and names of other animations associated with that tag. Changes in the sound or image do not produce any immediate change in the current animation or sound playing in that IAVO (unless it occurs in the rectangle relative to the loop that is playing). In the left of the rectangle is a play button, which previews the current selection of sound and image, without closing the "back" of the object. The play button becomes then a stop button.

When the user presses the central button, the "back" of the IAVO is closed, with an animation that mirrors its opening (in reverse). If there were any changes in the sound or animation, within the rectangle relative to the loop that was playing before, these changes will now be reflected. In case the preview had been active previously to closing, that audiovisual loop remains playing. If the IAVO was stopped, it will remain stopped, unless the preview had been active.

4.7 Deleting an IAVO, and Making it Reappear

When an IAVO is stopped, its red button is not longer a stop button, but instead a delete button (as was shown in Figure 1.). When this delete button is pressed, the object disappears, and the correspondent colored bar reappears, in the same position it had in the start of the application.

This reappearance occurs with an animation: the bar appears from the top and bottom edges of the stage. By clicking anywhere in this bar, the bar disappears again, and the IAVO reappears in that point (stopped). The bar disappears into the top and bottom edges of the screen by splitting in two at the point where the user has clicked (Figure 6.).

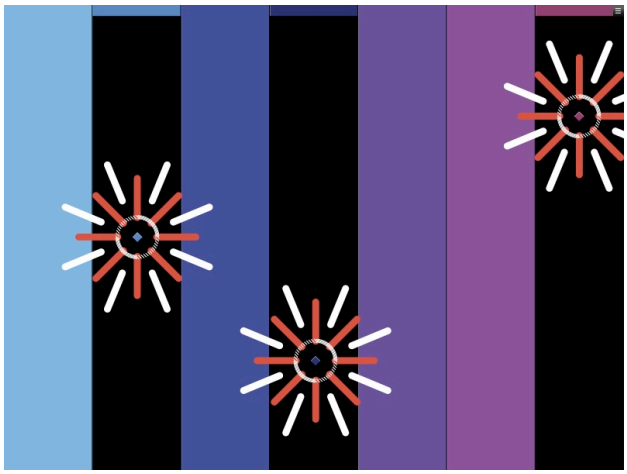


Figure 6. Stage with four active bars, representing four deleted IAVOs

4.8 "Back Stage"

In the upper right corner of the stage, there is a "back stage" button. Activating the "back stage" option allows for introducing changes to multiple objects in the same screen, and also for changing the tag of each IAVO (Figure 7.).

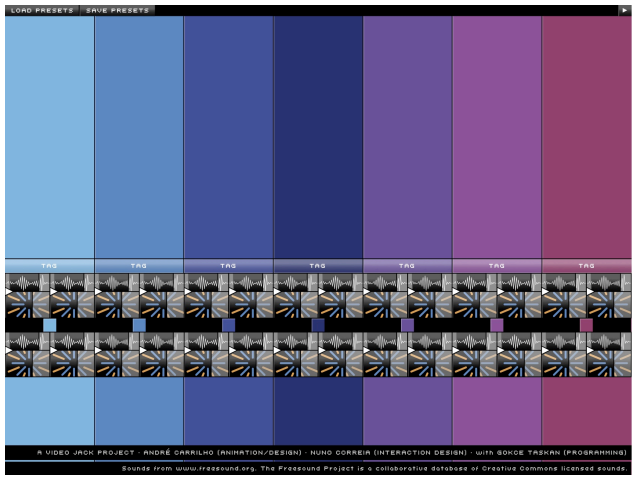


Figure 7. "Back stage" of *AV Clash*

Pressing the "back stage" button triggers a series of events. The "back stage" button becomes a "stage" button. If any object had its "back" visible, it will be closed first. All IAVOs move back to the area of their original bar, aligned vertically to the same position, near the bottom of the stage. All bars of active IAVOs reappear by expanding from the top and bottom of the stage to the vertical position of the IAVO, but do not close completely, stopping at the edge of its ring. Regarding the IAVOs that were not active, their bars open slightly, revealing their ring (aligned vertically with the remaining objects). Immediately after, all IAVOs reveal their "back", with the same animation described above. One more element is shown in this animation: a tag button appears above the objects, sliding from behind them. The sounds and animations that were playing previously continue to play. Two black bars appear on top and in the bottom of the screen. The top bar reveals load and save

options, while the bottom bar shows the credits of the project.

Now users can change all animations and sounds of all IAVOs, similarly to how they could change the "back" of each IAVO individually. Additionally, users can change tags. By pressing the tag button that is now on top of each IAVO, a pop-up appears with a list of tags. (Figure 8.).

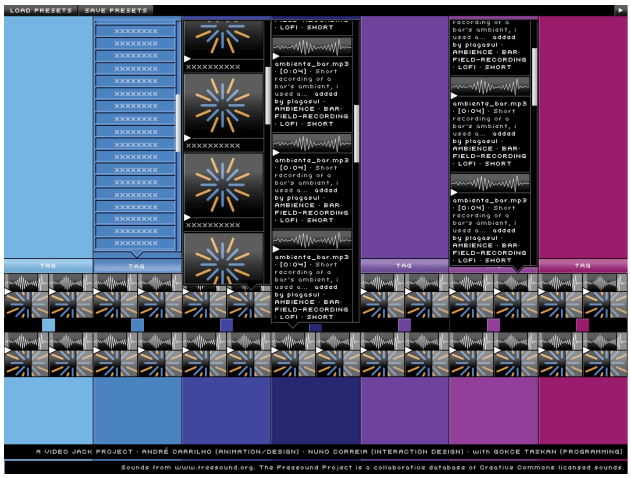


Figure 8. "Back stage" with several pop-ups open

A change in tag loads four new sounds from *Freesound.org*, chosen randomly among the 20 most popular ones with that tag. A pre-loader animation then takes place, similar to the circular initial pre-loader, and the sound change pre-loader (overlapped with the IAVO). When the loading process concludes, all sounds and animations of the IAVO are replaced. The IAVO reverts to its defaults, and if any sound or animation was playing, it is stopped. The bars disappear, mirroring the start of the project.

The "Save Set" button generates a file storing all the options for each IAVO: tag and list of the four sounds and animations; volume and effect level information; and audio and visual effect. The "Load Set" button loads a previously recorded file, consequently changing all the information of each IAVO, and loading new sounds.

5. REFLECTIONS ON THE INTERACTION DESIGN OF *AV CLASH*

In *AV Clash*, only the most relevant UI elements are visible at a given time. Since IAVOs contain a large amount of buttons and interactive elements, they are shown and hidden depending on the position of the mouse relative to the IAVO, and if the IAVO is playing or not. For example, rolling over the ring of a playing IAVO reveals the interactive possibilities contained in the ring, previously hidden (volume and effect fader thumbs, enlarged ring for dragging and throwing). If the user moves the cursor further towards the inside of the ring, more interface options appear (playback and effects buttons). Donald Norman classifies this approach as modularization - creating separate functional modules, "each with a limited set of controls, each specialized for some different aspects of the task" [5, p. 174].

Visibility is important not only for modularity, but also to hide irrelevant options in a certain context. For example, if an IAVO is not playing, the volume and effect faders are hidden. As Donald Norman states, "a good designer makes sure that appropriate actions are perceptible and inappropriate ones invisible" [5, p. xii].

This hiding and showing of elements also indicates feedback, sending users "information about what action has actually been done, what result has been accomplished" [5, p. 27]. Another example of this principle occurs when users press one of the playback selection buttons – the button becomes invisible.

The graphic design of interactive elements in *AV Clash* is meant to convey its functionality. An example of this approach is the design of the ring, which has a jagged appearance, meant to reflect its "draggable" affordance. According to Norman, affordances refer to "the perceived and actual properties of the thing, particularly those fundamental properties that determine just how the thing could possibly be used" [5, p. 9].

The notion of mapping, meaning the relationship between the controls and the results [5, p. 23], is also explored in *AV Clash*. An example of this is the mapping of each IAVO to a specific screen area in the beginning of the session, to which it returns to when users press the "back stage" option. Mappings are also implemented when a playback selection button expands to a full audio and visual loop selection interface – the audio and visual loop selection options are located in the same quadrant of the correspondent selection button.

Besides an emphasis on functionality, the UI of *AV Clash* also references its own aesthetic universe, connected to the circular nature of the animations. These animations, although abstract, are often inspired by concentric "real" objects such as flowers, planets, biological and molecular/atomic structures. The visual appeal of the UI is meant to induce playfulness: "the mouse and the pen-based interface allow the user the immediacy of touching, dragging, and manipulating visually attractive interfaces" [1, p. 23].

6. CONCLUSIONS

6.1 Assessment Regarding Initial Aims

The author considers that *AV Clash* has fulfilled the objective of developing the structure behind *AVOL* into a more flexible project, in terms of source sounds (imported from *Freesound.org*), animations (although less diverse than the sounds) and audiovisual manipulation (through the implementation of sonic and visual effects). The introduction of the "throw" behavior and the development of the "clash" behavior contributed to a higher degree of playfulness. The use of colors and tags (imported from *Freesound.org*) to identify IAVOs also facilitates recognition of objects. The colored bars create a higher visual diversity on the stage. The "save set" functionality introduces an option to save all settings for all IAVOs, allowing for the storing and sharing of user op-

tions. Besides loading new sounds, the "load set" button can quickly change multiple parameters within a session.

However, several limitations have been identified by the author in the project, which should be addressed in a future project, or in a new version of *AV Clash*.

6.2 Paths for future developments

AV Clash could benefit from accessing more content, and from having more content manipulation capabilities. The visual diversity of *AV Clash* is still small compared to its audio side. A database for visuals (possibly vector based animations) could be created, which would then be used by *AV Clash* similarly to *Freesound.org* for sound. Further audio and visual effects could be added. Specific tags for *AV Clash* could be created in *Freesound.org* to ensure coherence of results.

Recording capabilities could be built in *AV Clash*, in order to allow users to record their sessions. Content sharing could also be implemented, not only of sets but also of those recording sessions. This content sharing could also integrate with profiles of users in *Freesound.org*.

Collaboration functionalities could also be implemented. Users could be allowed to take control of a certain IAVO, and play with it. The different users would "jam" together, each with his/her own IAVO. This system would resemble how a band plays in a "real life" stage.

A "sequence" mode could be implemented in *AV Clash*, which would not simply playback one of the sounds of each IAVO, but instead run through the four sounds – in a linear sequence; randomly; or in another sequence defined by the user (for example, by drawing a path in the "back" of the object connecting the four sounds, thus specifying the order). This "sequence" mode is inspired by *Freesound Radio*.

6.3 Final Reflections

These conclusions are preliminary. *AV Clash* is under development, as mentioned before, and is being tested by Video Jack. Therefore it is not online yet, and no public presentations have been done. Video Jack intend to release *AV Clash* online soon, in July 2010. They wish to start presenting the project shortly after, as installation and performance. A domain name has been registered for the project.[3] Currently it only displays a few preliminary recordings using a prototype version of the project.

Feedback gathered from users after releasing *AV Clash*, and from presentations, will enrich the conclusions presented in this paper. The author intends to conduct interviews to users, to better assess these conclusions.

The author believes that there is a vast potential for the type of application that *AV Clash* represents – a playful tool for integrated audiovisual expression, which gathers audiovisual resources from Internet repositories, and that explores the potential of connecting users through the Web via their creativity.

[3] http://www.avclash.com

7. REFERENCES

[1] J. D. Bolter and R. Grusin: *Remediation: Understanding New Media*, MIT Press, Cambridge, 2000.

[2] S. Jordà: *Digital Lutherie - Crafting Musical Computers for New Musics' Performance and Improvisation*, PhD Thesis, Universitat Pompeu Fabra, Barcelona, 2005. http://www.iua.upf.es/~sergi/tesi/ Referenced April 7 2010.

[3] W. Moritz: "Color Music – Integral Cinema", in *Poétique de la Couleur*, Musée du Louvre, Paris, 1995. http://www.centerforvisualmusic.org/WMCM_IC.htm Referenced 7 April 2010.

[4] W. Moritz: "The Dream of Color Music and the Machines that Made it Possible", *Animation World Magazine*, Apr. 1997. http://www.awn.com/mag/issue2.1/articles/moritz2.1.html Referenced 7 April 2010.

[5] D. Norman: *The Design of Everyday Things*, Basic Books, New York, 2002.

[6] C. Paul: *Digital Art*, Thames & Hudson, London, 2003.

[7] G. Roma, P. Herrera and X. Serra: "Freesound.org Radio: supporting music creation by exploration of a sound database", in *Proceedings of Computational Creativity Support Workshop CHI09*, 2009. http://mtg.upf.edu/node/1255 Referenced 20 April 2010.

[8] M. Tribe and R. Jana: *New Media Art*, Taschen, Köln, 2007.

[9] C. Van Campen: *The Hidden Sense: Synesthesia in Art and Science*, MIT Press, Cambridge, 2008.

PHOXES - MODULAR ELECTRONIC MUSIC INSTRUMENTS BASED ON PHYSICAL MODELING SOUND SYNTHESIS

Steven Gelineck and Stefania Serafin
Aalborg University Copenhagen
Department of Architecture, Design and Media Technology
stg@media.aau.dk

ABSTRACT

This paper describes the development of a set of electronic music instruments (PHOXES), which are based on physical modeling sound synthesis. The instruments are modular, meaning that they can be combined with each other in various ways in order to create richer systems, challenging both the control and perception, and thereby also the sonic potential of the models. A method for evaluating the PHOXES has been explored in the form of a pre-test where a test subject borrowed the instrument for a period of 10 days. The longer test period makes way for a more nuanced qualitative evaluation of how such instruments might be integrated into workflows of real world users.

1. INTRODUCTION

The PHOXES (Physical Boxes) are a set of musical instruments, which are based on physical modeling sound synthesis. They were developed in order to investigate how high level exploratory control structures have an impact on the sonic potential of physical models.

1.1 Exploring Physical Modeling

Traditionally the goal when developing physical models has been to accurately simulate the physical mechanisms, which produce sound in the real world. When controlling these models the goal has often been to achieve the same nuanced input capabilities as one would have when playing real acoustic instruments striving for an enhanced expressivity or intimacy.

This research deals with the ongoing investigation into how control structures for physical modeling sound synthesis, can enhance the explorability and thereby the creative potential of the technique. One goal is to understand how physical modeling can be controlled in order to accommodate the work processes of the end user (for us the experimental electronic musician). The focus is not on expressivity or intimacy of musical controllers, but on enhancing their exploratory and creative potential (note that these are not apposed to each other as a higher level of intimacy can also lead to a higher degree of exploration [1]).

Figure 1. The PHOXES system is modular and currently implements four different modules each implementing a different physical model and a different excitation controller. *Upper from left: friction PHOX and particle PHOX. Lower from left: drum PHOX and tube PHOX.*

We believe that physical modeling bears with it an obvious potential to maintain the balance between intuitive control and that certain amount of complexity that is needed in order facilitate the exploratory processes, which are so important for supporting creativity. Within creativity support tools research this balance is referred to as *Low threshold, high ceiling, and wide walls* [2].

One way of creating a low threshold can be to design input devices, which are built upon traditional acoustic instruments. Controls will not only be familiar, there will also be a great amount of users, who have already spent years on refining expert playing techniques. In developing the PHOXES we have worked in the opposite direction by leveraging on input devices and control structures found in commercial electronic music instruments, merging them with alternate input devices specifically suited towards the physical models.

2. PHOXES

Each PHOX is an instrument on its own implementing a physical model, an excitation controller and four knobs for adjusting various model parameters (mostly resonator parameters). The excitation controller lets the user inject energy into the physical model by performing musical gestures, which intuitively relate to that model. For instance the tube PHOX implements a flute controller for exciting a

turbulence model. The user receives visual feedback in the form of exact control values on an LCD screen mounted on each PHOX.

Each PHOX works as a musical instrument on its own, but the PHOXES are modular, meaning that two or more PHOXES can be combined in various ways to produce sonically richer systems. Although each physical model is still fixed this lets the user explore the models in a totally different and more abstract way. Each PHOX still upholds an intuitive perception of how the sound is produced, because of the perceived causality inherent in the physical modeling technique. But when they are combined this perceived causality is challenged, altering both the gesture space provided by the PHOXES and the sonic potential of the models. How this is handled is described later in Section 2.5.

The goal when developing the PHOXES was to create a flexible system that while keeping each physical model fixed (not letting the user assemble their own physical model as seen in for example [3]), the users are able to combine the different models in various ways thereby achieving a different exploration of the sonic possibilities made available by each model. This section will describe the design and implementation of the PHOXES - in particular the physical models used, the choice of control devices (including how they were built) and the mapping strategies for developing the modular system.

2.1 Physical Models

Each of the individual PHOXES implements a different physical model, each representing a different physical modeling technique. They vary in complexity, sonic fidelity and physicality (which type of excitation gesture they naturally propose). The four PHOXES (as seen in Figure 1) and the physical models on which they are based are:

- **tube PHOX** - implements a *turbulence model* with a simple nonlinear exciter [4] and a one-dimensional waveguide resonator [5].
- **particle PHOX** - implements a *particle model* with a stochastic excitation based on Physically Informed Sonic Modeling (PhISM) by Perry Cook [6].
- **friction PHOX** - implements a *friction model* with a complex nonlinear exciter [7] and a one-dimensional waveguide resonator.
- **drum PHOX** - implements two identical *drum models* each with a simple nonlinear exciter and a two-dimensional waveguide resonator [8].

2.2 Physical Devices

As described in Section 1.1 the PHOXES have been inspired by commercial electronic music instruments. It was important that the eventual test environment was as natural for the test subjects as possible, which is also why the PHOXES were designed with a look and feel that were convincing enough to resemble real commercial hardware synthesizers. The PHOXES could have been presented

Figure 2. The flute controller is implemented using an amplified low pressure sensor mounted to the end of a tube, which the user blows into.

(and perhaps partially controlled) in a software environment, but it was important for us to put emphasis on the physical devices as standalone instruments - even though they are not. Finally, it was crucial that they were robust and durable enough to make a long term evaluation possible.

Each of the four PHOXES is implemented using a PhidgetTextLCD with PhidgetInterfaceKit 8/8/8 [1], which provides 8 analog inputs, 8 digital inputs, 8 digital outputs, and a 2-line by 20-character LCD screen. This makes it possible to control mapping settings, control settings, and display settings in a customized menu system directly on each of the instruments. The instruments connect to the computer via USB and communication, sound synthesis and mapping is handled directly from Max/MSP. The system has been tested on a MacBook Pro with 2.4 GHz Intel Core 2 Duo processor and 4GB 667 MHz DDR2 SDRAM - Mac OSX 10.5.8.

2.3 Excitation Controllers

Each PHOX implements a different excitation controller, which naturally relates to the physical model of that PHOX. The excitation controllers are as follows:

2.3.1 tube PHOX Excitation Control - Flute

The tube PHOX implements a flute controller, which by default controls the turbulence model. The flute controller implements an amplified low pressure sensor [2], which is attached to a tube that the user blows into - see Figure 2. The pressure sensor is very responsive and is sensitive enough for detecting very small differences in air pressure produced by the blowing gesture and because the signal is amplified it connects directly into the Phidget interface [3]. The air pressure is mapped to the input energy into the physical model.

2.3.2 particle PHOX Excitation Control - Crank

The particle PHOX implements a crank as its default excitation controller - see Figure 3. The crank is attached to a

[1] from http://phidgets.com
[2] the 1INCH-D-4V from All Sensors
[3] could also be an Arduino or CUI interface or the likes

Figure 3. The crank is used as excitation controller for the particle PHOX. It is attached to a multi-turn rotational potentiometer.

Figure 4. The friction PHOX implements a ribbon sensor, which lets the user slide his or finger back and forth over the surface to create energy.

multi-turn rotational potentiometer [4] and the rotational velocity of the potentiometer is mapped to the input energy of the physical model - the probability of a particle hit in the case of the particle PHOX.

2.3.3 friction PHOX Excitation Control - Slide Surface

The friction excitation controller is implemented using a ribbon sensor (a soft potentiometer [5]) - see Figure 4. The user slides his or her finger back and forth on the surface to create energy. The velocity of the motion is mapped to the input energy of the physical model.

2.3.4 drum PHOX Excitation Control - Drum Triggers

The excitation controller for the drum PHOX consists of two drum triggers built from piezo transducers [6] - see Figure 5. The transducer produces a voltage when struck - this is thresholded to detect a hit and peak detected to determine the velocity of the hit.

2.4 Model Parameter Controls

Each PHOX also implements four knobs, which let the users control selected parameters of the physical model. For instance one is able to adjust how long the tube is, or how dampened the particles collide. Physically, they are controlled by simple knobs (potentiometers), which help establish the look of the PHOXES by aesthetically connecting them to more traditional electronic instruments or controllers. They present a familiar control surface, which lowers the threshold for electronic musicians initially learning the instruments and finally, because they are the same on each PHOX, they help to perceptually connect the different PHOXES into one system.

[4] Model 357 from Vishay
[5] SoftPot from Spectra Symbol
[6] KPSG100 from Kingstate

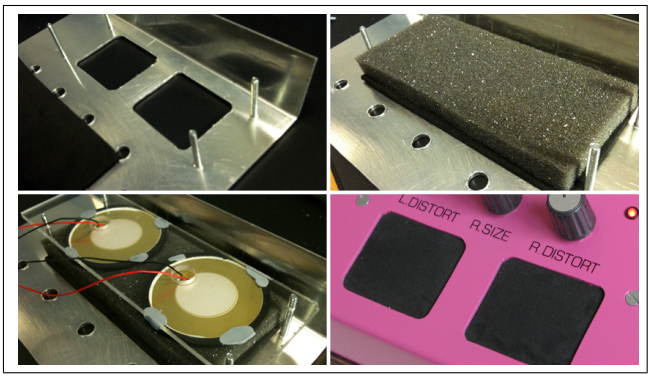


Figure 5. The drum PHOX implements two drum triggers, which were implemented by mounting two piezoelectric discs under two layers of foam.

The following is an overview of which model parameters are controllable. Parameters for the tube PHOX are *tube length 1, tube length 2, vibrato*, and *flute airyness)*. Parameters for the particle PHOX are *fundamental frequency, approximate frequency of four partials, amount of randomization of partial frequencies*, and *bandwidth of the partials*. Parameters for the friction PHOX are *frequency 1, frequency 2, downward force*, and *roughness (randomness of force and amount of noise)*. Finally, parameters for the drum PHOX are *left drum size, left drum frequency distortion, right drum size*, and *right drum frequency distortion*.

2.5 Modularity

2.5.1 Exploration of excitation gestures

By default each PHOX has a dedicated controller, which is intended to presents a natural intuitive relationship between excitation gesture and model. This helps the user to get a first intuitive impression of the model's control possibilities. However, the user can also choose to control the physical model using the excitation controller imbedded in any of the other PHOXES. For instance instead of exciting the friction model of the friction PHOX using the *slide surface* one is able to use the *crank*. This lets the user explore different playing styles by performing different excitation gestures - thereby hopefully achieving a deeper exploration of the sonic potential of the physical models.

The flexibility of the PHOXES system entails an implementation challenge because each PHOX must uphold a meaningful relationship between input gesture and the sound being produced no matter what type of excitation gesture. The idea is to use *energy* as the common denominator as each model relies on energy in order to be excited. But how that energy mechanically relates to each model must be defined. The challenge becomes particularly interesting when shifting between continuous excitation gestures (e.g. blowing into the flute controller) and instantaneous excitation gestures (e.g. tapping/striking the drum trigger). A number of different possible mapping solutions were considered, but we chose to map the energy of a drum hit to an energy envelope, which has a peak proportional to the hit velocity and which decays in energy again proportional to the hit velocity (linear decay lasting

between 200 and 500 ms.). This means that when using the drum triggers to excite for instance the turbulence model, the amount of air pressure (exciting the turbulence model) will be enveloped according to the hit velocity of the drum.

For mapping a continuous gesture (e.g. rotating the crank) to a model that normally is excited by instantaneous gestures (tapping/striking the drum trigger) a similar challenge occurs. We have chosen to let the instantaneous gesture take shape as a scraping mechanism, which creates small instantaneous excitations we can use for exciting the drum. How frequent the excitations occur and their individual velocities depend on the velocity of the continuous gesture.

2.5.2 Controlling one physical model with another

Energy into the physical model of a PHOX need not come from an excitation controller. The system makes it possible for the user to drive the physical model using the output sound from a different model - similar to [9]. This means that for instance the turbulence model, which by default is excited with a certain amount of white noise (proportional to how hard the user blows), can be excited by the output sound from e.g. the drum model. This is done by substituting the white noise with the audio output from the drum model. It thus becomes the drum sound, which drives the turbulence model. The result is a sort of fusion between the two models, where the turbulence model acts as a sort of audio effect, which is used to color the drum sound.

Earlier research has shown that interesting timbres from one model can be transferred to another model, and models, which users rate as boring can become interesting when combined in this fashion with other models - (even with each other) [10]. The PHOXES extend this idea by making it possible to combine many models at the same time (for instance use the crank of the particle PHOX to excite the turbulence model of the tube PHOX then letting the resulting audio signal excite the friction PHOX and so on and so forth).

Because the user is able to excite one PHOX with audio output from a different PHOX, each model must have a way of taking audio as input and somehow substituting that with the energy input of the model.

For complete details regarding mapping go to `http://media.aau.dk/~stg/phoxes/`.

3. PRE-TEST

In order to explore a suitable method for evaluating the PHOXES a pre-test was conducted. Carrying out any formal evaluation of these kinds of instrumental systems in the rather complex environment of creative music making has proven to be quite challenging. Different evaluation methods have been proposed for evaluation of musical interfaces inspired by methodologies found in the field of Human Computer Interaction (HCI) [11, 12]. For this pre-test we wanted to explore methodologies related not so much to the performance or usability of the system (how well the user is able to perform specific tasks) but more the overall experience with the system dealing with softer hedonic qualities [13, 14, 15] - for instance how well the user identifies with the instruments, whether they are inspiring to work with or how well the system supports musical exploration.

Most formal evaluations of musical interfaces are carried out under circumstances far from the natural environment of the electronic musician, which may be adequate for various specific usability issues [16]. But we believe that this makes it difficult to evaluate factors of more qualitative nature. Earlier research [17] has also suggested that tests need to be carried out over longer periods of time, which is especially enforced when evaluating more complex systems.

The pre-test was carried out using one male test person who is an experienced experimental electronic musician. He has extensive experience with both traditional acoustic instruments (mostly percussion instruments) and with various electronic instruments as both a composer and a performer. The test took place over a period of 10 days where the test-subject borrowed the PHOXES. The test was very free as the test person did not receive any instruction as to any specific tasks to perform during the 10 days. The test subject was instructed to treat the instruments as he would any new musical device that came into his possession.

In order to asses the implications of the longer test period, first impressions were noted by having the test subject fill in a questionnaire after having played around with the PHOXES for approximately one hour. The questionnaire was comprised of two forms: One was the AttrakDiff [7] hedonic / pragmatic evaluation form also used in [13], which lets the user rate the system based on a series of opposite/bipolar word-pairs relating to hedonistic and pragmatic qualities of interactive systems. The other was a semi-quantitative Likert-scale style evaluation form that lets the user rate each individual PHOX and the overall system in regards to features more closely related to the specific area of physical modeling based electronic instruments - such as whether the instruments provided sonic diversity, or felt like a real acoustic instrument. The same questionnaire was filled out after the 10 days test period. Finally an open interview was conducted to gather qualitative statements about how the test-subject worked with the PHOXES, whether that changed throughout the test-period and which issues arose during the test-period.

4. RESULTS AND DISCUSSION

The main focus when evaluating the data collected in the pre-test was on the methodological approach, and specifically on what kind of system improvements have to be made if this kind of evaluation method is to succeed on a greater scale. However, the results of the evaluation have been included to provide an initial idea of the perceived qualities of the PHOXES system. Note that they are totally subjective and inconclusive as only one test subject participated in the test.

The results indicate that the test subject was highly motivated and stimulated by the PHOXES system (Hedonic Quality - Stimulation) and found them having a high per-

[7] http://www.attrakdiff.de

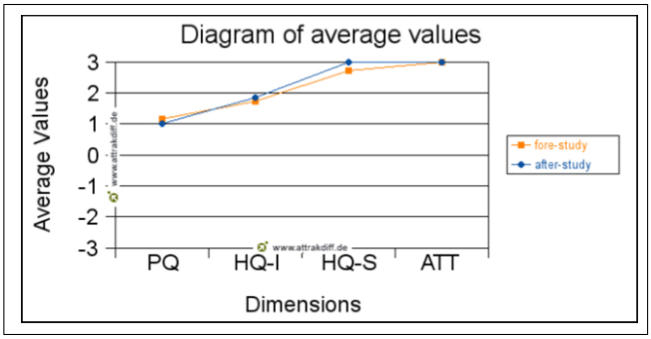


Figure 6. Results of the AttrakDiff evaluation. The following dimensions are evaluated: Pragmatic Quality (PQ), Hedonic Quality - Identity (HQ-I), Hedonic Quality - Stimulation (HQ-S) and Attractiveness (ATT). Fore study corresponds to first impressions and after study corresponds to the final evaluation.

ceived quality (Attractiveness). The subject's identification with the system was above average (Hedonic Quality - Identity) - as so was the perceived usability (Pragmatic Quality).

Surprisingly the perceived hedonic and pragmatic qualities stayed more or less unaltered when comparing answers from the first impressions evaluation and the final evaluation after the 10 days - See Figure 6.

Problems with the PHOXES in regards to the relatively uncontrollable test scenario were found in the computational cost of the physical models. The DSP CPU load would limit the test subject as he integrated the PHOXES into larger sequences/multitrack recordings in his preferred digital audio workstation (Ableton Live). This was quite unfortunate, as it is important for us to examine how the PHOXES are able to integrate into the work flow of eventual future test subjects in order to evaluate their exploratory qualities. Apart from cleaning up the code (Max/MSP patch and externals) making it run more smoothly, a solution could be to keep the processing on a separate dedicated machine. On the positive side, the physical interfaces were easy to setup and physically durable enough for the 10 days test period.

There was a problem that the test subject did not get to explore parts of the modular system. As the test subject put it; he didn't get to the advanced settings. It is difficult to say whether the system was too complicated, whether the system was not presented intuitively enough, or whether the test period might have been too short. The subject might also have been too focussed on improving playing skills, focussing on the interplay between controllers and models, and not so much on the combining of models. On one hand more time or explicit tasks could be given to the participants in order to get them to focus on certain parts of the system. On the other hand it is valuable to see how different uses of the system might arise by absence of specific tasks.

We were pleased to experience that the PHOXES system was robust enough to handle 10 days of use without our interference. For future testing we will improve the PHOXES in accordance with the improvements described above. We will continue to explore the methodological approach, including a longer test period and more task oriented restrictions to parts of the evaluation period.

5. REFERENCES

[1] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music Journal*, vol. 26, no. 3, pp. 11–22, 2002.

[2] B. Shneiderman, G. Fischer, M. Czerwinski, M. Resnick, B. A. Myers, L. Candy, E. A. Edmonds, M. Eisenberg, E. Giaccardi, T. T. Hewett, P. Jennings, B. Kules, K. Nakakoji, J. F. Nunamaker, R. F. Pausch, T. Selker, E. Sylvan, and M. A. Terry, "Creativity support tools: Report from a u.s. national science foundation sponsored workshop," *International Journal of Human Computer Interaction*, vol. 20, no. 2, pp. 61–77, 2006.

[3] C. Cadoz, A. Luciani, and J. L. Florens, "Cordis-anima: A modelling and simulation system for sound and image synthesis — the general formalism," *Computer Music Journal*, vol. 17, no. 1, 1993.

[4] P. Cook, "A meta-wind-instrument physical model, and a meta-controller for real time performance control," in *Proceedings of the International Computer Music Conference*, pp. 273–276, San Francisco, California: International Computer Music Association, 1992.

[5] J. O. Smith, *Physical Audio Signal Processing, December 2008 Edition*. `http://-ccrma.stanford.edu/~jos/-pasp/Digital_Waveguide_Models.html`, 2008. online book.

[6] P. R. Cook, "Physically informed sonic modeling (phism): Synthesis of percussive sounds," *Computer Music Journal*, vol. 21, no. 3, pp. 38–49, 1997.

[7] F. Avanzini, S. Serafin, and D. Rocchesso, "Modeling interactions between rubbed dry surfaces using an elasto-plastic friction model," in *Proceedings of the International Conference on Digital Audio Effects*, (Hamburg), pp. 111–116, DAFX, 2002.

[8] S. A. V. Duyne and J. O. Smith, "Physical modeling with the 2-d digital waveguide mesh," in *Proceedings of the International Computer Music Conference*, pp. 40–47, San Francisco, California: International Computer Music Association, 1993.

[9] S. Gelineck and S. Serafin, "A practical approach towards an exploratory framework for physical modeling," *Computer Music Journal*, vol. 34, no. 2, 2010.

[10] N. Böttcher, S. Gelineck, and S. Serafin, "Physmism: A control interface for creative exploration of physical models," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2007.

[11] M. M. Wanderley and N. Orio, "Evaluation of input devices for musical expression: Borrowing tools from hci," *Computer Music Journal*, vol. 26, no. 3, pp. 62–76, 2000.

[12] C. Kiefer, N. Collins, and G. Fitzpatrick, "Hci methodology for evaluating musical controllers: A case study," in *Proceedings of NIME*, 2008.

[13] C. Geiger, H. Reckter, D. Paschke, F. Schutz, and C. Poepel, "Towards participatory design and evaluation of theremin-based musical interfaces," in *Proceedings of the Conference on New Interfaces for Musical Expression*, 2008.

[14] M. Hassenzahl, A. Platz, M. Burmester, and K. Lehner, "Past, present, and future of user interface software tools," *CHI Letters*, vol. 2, no. 1, pp. 201–208, 2000.

[15] D. Fallman and J. Waterworth, "Dealing with user experience and affective evaluation in hci design: A repertory grid approach," in *CHI 2005, Conference on Human Factors in Computing Systems*, 2005.

[16] M. T. Marshall, M. Hartshorn, M. M. Wanderley, and D. J. Levitin, "Sensor choice for parameter modulations in digital musical instruments: Empirical evidence from pitch modulation," *Journal of New Music Research*, vol. 38, no. 3, pp. 241–253, 2009.

[17] S. Gelineck and S. Serafin, "A quantitative evaluation of the differences between knobs and sliders," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2009.

INTERLUDE - A FRAMEWORK FOR AUGMENTED MUSIC SCORES

D. Fober C. Daudin Y. Orlarey S. Letz
Grame - Centre National de Création Musicale
{fober,daudin,orlarey,letz}@grame.fr

ABSTRACT

An *Augmented Music Score* is a graphic space providing the representation, composition and manipulation of heterogeneous music objects (music scores but also images, text, signals...), both in the graphic and time domains. In addition, it supports the representation of the music performance, considered as a specific sound or gestural instance of the score. This paper presents the theoretical foundation of the augmented music score as well as an application - an augmented score viewer - that implements the proposed solutions.

1. INTRODUCTION

Music notation has a long history and evolved through ages. From the ancient neumes to the contemporary music notation, the western culture is rich of the many ways explored to represent the music. From symbolic or prescriptive notations to pure graphic representation, the music score has always been in constant interaction with the creative and artistic process.

However, although the music representations have exploded with the advent of computer music [1, 2, 3], the music score, intended to the performer, didn't evolved in proportion to the new music forms. In particular, there is a significant gap between interactive music and the static way it is generally notated: a performer has generally a traditional paper score, plus a computer screen displaying a rough number or letter to indicate the state of the interaction system. At the same time, we can observe the emergence of new needs in terms of music representation.

In the domain of electro-acoustic music, analytic scores - music scores made *a postriori* - like the "Portraits polychromes" [1] , become common tools for the musicologists but have little support from the existing computer music software, apart the remarkable approach proposed for years by the Acousmograph [4, 5].

In the music pedagogy domain and based on a mirror metaphor, experiments have been made to extend the music score in order to provide feedback to students learning and practising a traditional music instrument [6, 7]. With this approach, an extended music score has been developed, supporting various annotations, including performance representations based on the audio signal, all in the context of a dynamic score layout. The limitations of the system rely mainly on a monophonic score centered approach and on a static design of the performance representation, making the system tricky to extend and to reuse.

New technologies allow now for real-time interaction and processing of musical, sound and gestural information. The Interlude project [2] takes place in this domain and touches upon new digital paradigms for exploration and interaction of expressive movement with music. The present work on *Augmented Music Scores* is part of this project and addresses interaction with symbolic content issues, while extending and generalizing previous music score extension approaches [7].

At the heart of the Augmented Music Score are the following main objectives:

- the score extension to arbitrary graphic objects: actually, we aim to consider arbitrary graphic objects (music scores but also images, text, signal representation...) as possible score candidates;
- the expression of relations between graphic and time space: considering that time is a constant and common property of all musical objects, we give a time position and a time dimension to any score component, which also implies to make the temporal relations graphically visible;
- the performance representation, gesture or audio based, with the aim to develop a system dynamically extensible.

None of the existing systems for music representation includes these kind of features: although generally extended to support contemporary music, tools like Lilypond [8], ENP [9], NoteAbility [10] or the Guido Engine [11] proposes a *traditional* music score approach and are not suited to dynamic music notation.

Although the organization of the graphic space consistently to the time space [12], or the synchronization of various medias [13], are among the current concerns, no formalism has been proposed to express relations between time and graphic space in a general music context.

Although tools for sound visualizations have been developed [14, 4], they are based on a fixed set of representations and don't support dynamic extensions.

[1] http://www.ina-entreprise.com/entreprise/activites/recherches-musicales/portraits-polychromes.html

[2] http://interlude.ircam.fr

Synchronization of heterogeneous medias in the graphic domain raises issues related to non-linearity, non-continuity, non-bijectivity. Based on previous experience, we propose to approach the problem using *segmentation* and the description of relations between *segments* - we will next use the term *mappings* to refer to these relations.

The representation of the music performance, whether sound or gestural, is approached with a reverse perspective: the graphic representation of a signal is viewed as a *graphic signal*, i.e. as a composite signal including all the information for the graphic rendering. This approach, which abstracts the representation calculation, results in an opened system, that can be dynamically extended.

We will first describe the theoretical foundations of the mappings and the context of use in the augmented music score framework. Next, we will explain how the system handles the performance signals to build *graphic signals*. Finally, an augmented music score viewer is presented and particularly its control API, which is actually a set of OSC messages[15].

2. TIME AND GRAPHIC RELATIONS

We talk of *time synchronization in the graphic domain* to refer to the graphic representation of the temporal relations between components of a score. Our previous experience in this domain [6, 7] led us to approach the question of these relations by the means of *segmentation* and the description of relations between *segments*. The term *mappings* is used to refer to these relations.

The role of a *mapping* is to make connections between the *segments* of different resources, where a *segment* is a contiguous zone of a resource. As previously mentioned, a resource can be an arbitrary object (music notation, images, text, signals...). The mappings are typically used to link graphic positions, music time and audio resources locations. For example, a mapping between an audio recording and a music score makes the connection between audio locations (expressed in frames) and the music time (expressed in quarter note divisions). A mapping between a music score and its graphic representation makes the connection between music time positions and graphic positions. Combining these mappings allows to make connections between all the time based resources.

The next sections describe the theoretical foundation for the notions of *segment, segmentation* and *mapping*. These foundations are independent of any implementation and of any resource specific information. They are followed by concrete use cases, implemented in the framework of the augmented score viewer.

2.1 Definitions

We will first introduce the notions of time and graphic segments. Next we will generalize these concrete definitions to an abstract, generic segment definition.

2.1.1 Time segment

A time segment is defined as an interval $i = [t_0, t_1[$ such as $t_0 \leqslant t_1$.

An interval $i = [t_0, t_1[$ is said empty when $t_0 = t_1$. We will use $\oslash$ to denote empty intervals.

Intersection of time segments (figure 1) is the largest interval such as:

$$\forall i_m,\ \forall i_n,\ i_m \cap i_n := \{j \mid j \in i_m\ \wedge\ j \in i_n\}$$

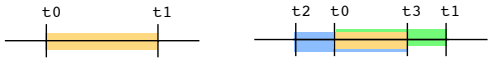


Figure 1. From left to right: a time segment and time segments intersection.

2.1.2 Graphic segment

A graphic segment g is defined as a rectangle given by two intervals $g = (i_x, i_y)$ where i_x is an interval on the x-axis and i_y, on the y-axis.

A graphic segment $g = (i_x, i_y)$ is said empty when $i_x = \oslash$ or $i_y = \oslash$

The intersection operation $\cap$ between graphic segments (figure 2) is defined such as:

$$\forall g = (i_x, i_y),\ \forall g' = (i'_x, i'_y),\ g \cap g' = (i_x \cap i'_x, i_y \cap i'_y)$$

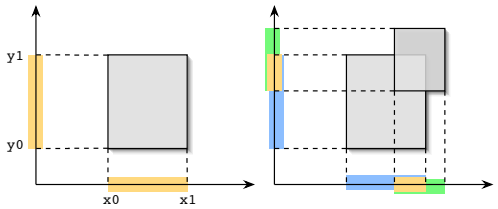


Figure 2. From left to right: a graphic segment and graphic segments intersection.

2.2 Segment generalization

We will extend the definitions above to a general definition of a n-dimensional segment. A n-dimensional segment is defined as a set of n intervals $s^n = \{i_1, ..., i_n\}$ where i_j is an interval on the dimension j.

A segment s^n is said empty when $\exists i \in s^n \mid i = \oslash$

Intersection between segments is defined as the set of their intervals intersection:

$$s_1^n \cap s_2^n = (i_1 \cap j_1, ..., i_n \cap j_n) \qquad (1)$$

where $s_1^n = (i_1, ..., i_n)$ et $s_2^n = (j_1, ..., j_n)$

2.3 Resource segmentation

A *segment-able* resource R is an n dimensions resource defined by a segment S^n of dimension n.

The segmentation of a resource R is the set of segments $Seg(R) = \{s_1^n, ...s_i^n\}$ such as:

$\forall i, j \in Seg(R)$	$i \cap j = \oslash$	the segments are disjoint
$\forall i \in Seg(R)$	$i \cap S^n = i$	segments are included in R

2.4 Mapping

A *mapping* is a relation between *segmentations*.

For a mapping $M \subseteq Seg(R_1) \times Seg(R_2)$ we define two functions:

$$M^+(i) = \{i' \in Seg(R_2) \mid (i, i') \in M\} \quad (2)$$

that gives the set of segments from R_2 associated to the segment i from R_1; and the reverse function:

$$M^-(i') = \{i \in Seg(R_1) \mid (i, i') \in M\} \quad (3)$$

that gives the set of segments from R_1 associated to the segment i' from R_2.

These functions are defined for a set of segments as the union of each segment mapping:

$$M^+(\{i_1, ...i_n\}) = \{M^+(i_1) \cup M^+(i_2) ... \cup M^+(i_n)\} \quad (4)$$

and

$$M^-(\{i_1, ...i_n\}) = \{M^-(i_1) \cup M^-(i_2) ... \cup M^-(i_n)\} \quad (5)$$

2.5 Mappings composition

Mappings composition is quite straightforward.
For a mapping $M_1 \subseteq Seg(R_1) \times Seg(R_2)$
and a mapping $M_2 \subseteq Seg(R_2) \times Seg(R_3)$, then :

$$M_1 \circ M_2 \subseteq Seg(R_1) \times Seg(R_3)$$

2.6 Augmented Score segmentations and mappings

All the resources that are part of an augmented score have a graphic and a temporal dimension. Thus, they are *segment-able* in the graphic and time spaces. Unless specified otherwise, time is referring to music time (i.e. metronomic time).

In addition, each resource type is *segment-able* in its specific space: audio frames linear space for an audio signal, two dimensional space organized in lines/columns for text, etc.

type	segmentations and mappings required
text	*graphic* ↔ **text** ↔ **time**
score	*graphic* ↔ *wrapped time* ↔ *time*
image	*graphic* ↔ **pixel** ↔ **time**
vectorial graphic	*graphic* ↔ **vectorial** ↔ **time**
signal	*graphic* ↔ **frame** ↔ **time**

Table 1. Segmentations and mappings for each component type

Table 1 lists the segmentations and mappings used by the different component types. Mappings are indicated using arrows (↔). Note that the arrows link segments of different types (the *segment* qualifier is omitted). Segmentations and mappings in *italic* are automatically computed by the system, those in **bold** have to be provided externally. This typology could be extended to any kind of resource, provided that for any new type, a mapping exists to go from the graphic space to the time space.

Note that an intermediate time segmentation, the *wrapped time*, is necessary for music score in order to catch repeated sections and jumps (to sign, to coda, etc.).

Composition of these mappings is at the basis of the mechanisms to address and synchronize the components both in the graphic and time spaces.

2.7 Synchronization examples

Let's consider two score components A and B with their corresponding graphic and time segmentations:

$$Seg(A_g), Seg(A_t), Seg(B_g), Seg(B_t).$$

In addition, B has an intermediate segmentation $Seg(B_l)$ expressed in the resource local space units (e.g. frames for an audio signal). The mappings

$M_A \subseteq Seg(A_g) \times Seg(A_t)$
and $M_B \subseteq Seg(B_t) \times Seg(B_l)$

give the correspondence between graphic and time space for A and between time and local space for B.

When synchronizing objects and deciding on what position should be used as base position, we have introduced a master/slave relation between components: a slave is always constrained to its master space.

2.7.1 Graphic alignment of time positions

It corresponds typically to the alignment of a cursor on a score: the cursor indicates a time position but without temporal extension.

Let's consider that B is A slave and we want to graphically align B to A at a time t. Let $s = [t_0, t_1[$ be the A segment containing the time t. The corresponding graphic segment is:

$$M_A^-(s) = \{g_i \in Seg(A_g) \mid (g, s) \in M_A\}$$

When $M_A^-(s)$ contains a single segment, B graphic position can be computed by simple linear interpolation i.e.:

$$(x_B, y_B) = (g_{x0} + (g_{x1} - g_{x0}).\delta,\ g_{y0})$$

where g_{x0} and g_{x1} are the graphic segment first and last x coordinates and $\delta = (t - t_0)/(t_1 - t_0)$.

y_B is arbitrary fixed to g_{y0} but it is actually controlled by a synchronization mode (over, above, below).

When s is mapped to several graphic segments, the operation can be repeated for each segment.

2.7.2 Segments graphic alignment

It corresponds typically to the alignment of a performance representation: the performance curve is made of segments (e.g. corresponding to notes) and each segment has to be aligned to the corresponding score location and duration.

The basic principle of segments alignment consists for each of the master graphic segment, to retrieve the corresponding slave segment expressed in the slave local coordinates and to render this slave segment in the space of the master graphic segment. Provided that $Seg(A_t) = Seg(B_t)$, the operation may be viewed as the mapping composition

$$M_A \circ M_B \subseteq Seg(A_g) \times Seg(B_l)$$

Figure 3 gives an example of different alignments obtained using different segmentations.

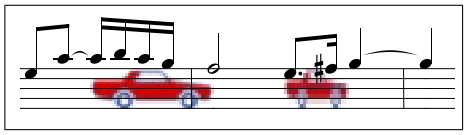


Figure 3. The same car bitmap synchronized to different time positions. The image is made of a single graphic segment and has a quarter note duration. It is stretched to the corresponding score graphic segments.

3. PERFORMANCE REPRESENTATION

Work on the performance representation takes root in previous experiences about the visualization of music instrument playing, made in a pedagogic context [7]. Our previous approach was based on a graphic rendering engine, taking signals and a representation type as input, and producing the corresponding image. The static embedding of the representation types in the rendering engine was one of the main limitations of the approach, implying to modify the engine for any new type.

In the context of the augmented score, our ambition was to develop a dynamically extensible system, avoiding this limitation. To do that, the graphic representation of a signal is viewed as a *graphic signal*, i.e. as a composite signal including all the information required for its graphic rendering.

The resulting performance representation object is a *first order* music score component: it has a date, a duration and thus can be synchronized to any other component.

3.1 Graphic signals

We define a graphic signal as a composite signal made of:

- a y signal: the graphic y coordinates
- a h signal: the graphic thickness at the y position
- a c signal: the graphic color

To make simple, we assume that the color space addressed by c has one dimension. Figure 4 gives an example of these parameters in the graphic space, at a time t.

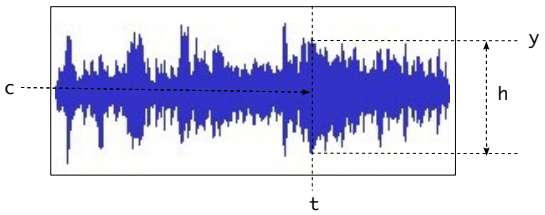


Figure 4. Graphic signal parameters at a time t.

Now, let's consider a signal S defined as a time function:

$$f(t) : \mathbb{R} \rightarrow \mathbb{R}^3 = (y, h, c) \mid y, h, c \in \mathbb{R}$$

then this signal may contain everything to be directly drawn i.e. without additional computation.

Such a system may also be viewed as an *oscilloscope* taking the 3 graphic signal components as input.

3.2 Signals composition

In order to build composite signals to be used as graphic signals, we have introduced a signals parallelization operation.

Let's consider $\mathbb{S}$, the set of signals $s : \mathbb{N} \rightarrow \mathbb{R}$.
The *parallel* operation '/' is defined as:

$$s_1/s_2/.../s_n : \mathbb{S} \rightarrow \mathbb{S}^n \mid s_i \in \mathbb{S} \quad (6)$$

The time function of a parallel signal $s^n \in \mathbb{S}^n$ is the parallelization of each signal's time function:

$$f(t) = (f_0(t), f_1(t), ...f_n(t)) \mid f_i(t) : \mathbb{N} \rightarrow \mathbb{R} \quad (7)$$

3.3 Parallel signals types

To implement the system, we have defined several parallel signals types:

- a *color signal* type, based on the HSBA color model [hue, saturation, brigthness, transparency]:

$$c ::= \overrightarrow{(h, s, b, a)} \mid h, s, b, a \in \mathbb{R}$$

- a *graphic signal* type that includes a y signal, a thickness signal th, followed by the 4 components of the color signal:

$$g ::= \overrightarrow{(y, th, h, s, b, a)} \mid y, th, h, s, b, a \in \mathbb{R}$$

- a *parallel graphic signals* type to support several graphic signals in parallel:

$$g^n ::= \overrightarrow{g} \mid g \in \mathbb{R}^6$$

3.4 Graphic signals examples

In order to validate our model, we will describe several representation types that were statically implemented with the previous approach.

3.4.1 Pitch representation

Represents notes pitches on the y-axis using the fundamental frequency (figure 5).

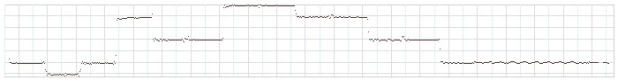


Figure 5. Pitch representation.

The corresponding graphic signal is expressed as:

$$g = S_{f0} \ / \ k_t \ / \ k_c$$

where S_{f0} : fundamental frequency
k_t : a constant thickness signal
k_c : a constant color signal

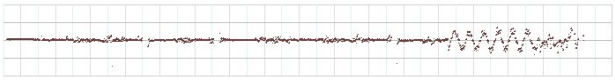


Figure 6. Intonation representation.

3.4.2 *Intonation representation*

Represents the difference between a fundamental frequency and a reference frequency (figure 6).

The corresponding graphic signal is expressed as:

$$g = S_{f0} - S_{fr} \ / \ k_t \ / \ k_c$$

where S_{f0} : fundamental frequency
S_{fr} : reference frequency
k_t : a constant thickness signal
k_c : a constant color signal

3.4.3 *Articulations*

Makes use of the signal RMS values to control the graphic thickness (figure 7).

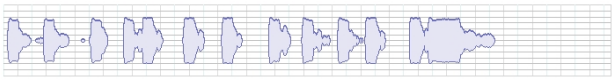


Figure 7. Articulations.

The corresponding graphic signal is expressed as:

$$g = k_y \ / \ S_{rms} \ / \ k_c$$

where k_y : signal y constant
S_{rms} : RMS signal
k_c : a constant color signal

3.4.4 *Pitch and articulation combined*

Makes use of the fundamental frequency and RMS values to draw articulations shifted by the pitches (figure 8).

Figure 8. Pitch and articulation combined.

The corresponding graphic signal is expressed as:

$$g = S_{f0} \ / \ S_{rms} \ / \ k_c$$

where S_{f0} : fundamental frequency
S_{rms} : RMS signal
k_c : a constant color signal

3.4.5 *Pitch and harmonics combined*

Combines the fundamental frequency to the first harmonics RMS values (figure 9). Each harmonic has a different color.

We will describe the corresponding graphic signal in several steps. First, we build the fundamental frequency graphic as above (see section 3.4.4) :

$$g0 = S_{f0} \ / \ S_{rms0} \ / \ k_c0$$

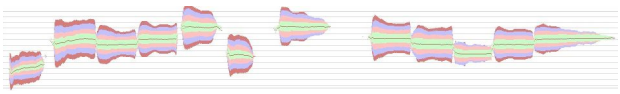


Figure 9. Pitch and harmonics combined.

where S_{f0} : fundamental frequency
S_{rms0} : f0 RMS values
k_c0 : a constant color signal

Next we build the graphic for the harmonic 1:

$$g1 = S_{f0} \ / \ S_{rms1} + S_{rms0} \ / \ k_c1$$

S_{rms1} : harmonic 1 RMS values
k_c1 : a constant color signal

Next, the graphic for the harmonic 2:

$$g2 = S_{f0} / \ S_{rms2} + S_{rms1} + S_{rms0} \ / \ k_c2$$

S_{rms2} : harmonic 2 RMS values
k_c2 : a constant color signal

etc.

And we finally combine them into a parallel graphic signal:

$$g = g2 \ / \ g1 \ / \ g0$$

4. THE AUGMENTED MUSIC SCORE VIEWER

The implementation takes the form of a C++ library - the Interlude library - as well as an augmented score viewer, build on top of this library. This viewer has no user interface since it has been primarily designed to be controlled via OSC messages i.e. using external applications like Max/MSP or Pure Data.

4.1 Messages general format

An Interlude OSC message is made of an OSC address, followed by a message string, followed by 0 to n parameters. The message string could be viewed as the method name of the object identified by the OSC address.

The OSC address is a string or a regular expression matching several objects. The OSC address space includes predefined static nodes:

`/ITL` corresponds to the Interlude viewer application
`/ITL/scene` corresponds to the rendering scene, actually the augmented score address.

The score components have addresses of the form:

`/ITL/scene/anyname` where `anyname` is an arbitrary user defined name.

The score components parameters can be addressed with messages strings like `x`, `y` or `z` to control the x, y or z position. Table 2 gives the main message strings, supported by all the components. Almost these messages have a relative form e.g. `dx` for a relative x displacement.

The next sections present examples of OSC messages setting up a score including synchronized components. Note that the messages list corresponds strictly to the file format of a score. Note also that these examples are static

message strings	component parameters
x, y, z, scale, angle	scale and position
date, duration, clock	time management
color, hsb	color management

Table 2. The main messages, supported by all the score components.

while real-time interaction is always possible, for example to move objects in time by sending `date` or `clock` messages (similar to MIDI clocks).

4.2 A simple cursor example

This example shows a cursor synchronized to a graphic bitmap. Lines beginning with a '#' are comments interleaved with the messages.

```
# creates the score as an image
/ITL/scene/turenas set img "score.png"
# sets the image graphic to time mapping
/ITL/scene/turenas mapf "turenas.map"
# sets the score title and position
/ITL/scene/title set txt "Turenas - John..."
/ITL/scene/title x -0.36
/ITL/scene/title y -0.86
/ITL/scene/title scale 3.0
# creates a rectangle used as cursor
/ITL/scene/cursor set rect 0.004 0.217176
/ITL/scene/cursor z 0.5
/ITL/scene/cursor color 204 0 48 132
# synchronizes the cursor to the score
/ITL/scene/sync cursor turenas v
# moves the cursor in time
/ITL/scene/cursor date 123 4
```

The mapping file *turenas.map* describes the relation betwwen the image segments and the time. The image is segmented in 3 parts corresponding to each lines. For each line, the first segment applies to the graphic space (expressed in pixels intervals) and the second segment to the time space (expressed as rationals).

```
( [27,780[ [15,193[ )   ( [0/4,225/4[ )
( [27,782[ [216,394[ )  ( [225/4,520/4[ )
( [27,511[ [417,594[ )  ( [520/4,594/4[ )
```

The result is given by the figure 10. The cursor is located to the image position corresponding to its date.

4.3 Nested synchronization example

This example uses 3 components; the first one is master of the second, which is master of the third one.

```
# creates a score using an image
/ITL/scene/score set img "score.jpg"
# sets the score graphic to time mapping
/ITL/scene/score mapf "score.map"

# creates a text using a text file
/ITL/scene/text set txtf "comment.txt"
# changes the text scale
/ITL/scene/text scale 3.0
# and the text color
/ITL/scene/text color 0 0 240 255
# put the text in front
/ITL/scene/text z 0.5
# and sets the text to time mapping
/ITL/scene/text mapf "comment.map"

# creates a ball as vectorial graphic
/ITL/scene/ball set ellipse 0.2 0.2
# puts it in front
/ITL/scene/ball z 0.4
# changes the ball color
/ITL/scene/ball color 250 50 0 255

# sets all the objects date
/ITL/scene/* date 4 1
# sets the text slave of the score
/ITL/scene/sync text score
# sets the ball slave of the text
/ITL/scene/sync ball text
```

Note the use of a wildcard in the OSC address to set all the objects date with a single message. The corresponding result is given by figure 11.

4.4 A signal synchronized to a score

This example show a graphic signal synchronized to a GMN score. Note that a graphic signal is a *first order* music score component: it has a date and a duration and can be synchronized to any other object.

```
# declare a y signal with size 200
/ITL/scene/signal/y size 200
# declare a thickness signal
/ITL/scene/signal/t size 200
# combines y and t + constant color signals
/ITL/scene/signal/sig set y t 0. 1. 1. 1.
# build the corresponding graphic signal
/ITL/scene/myGraph set graph sig
# set its date and duration
/ITL/scene/myGraph date 7 4
/ITL/scene/myGraph duration 2 4
# creates the score
/ITL/scene/score set gmnf "score.gmn"
# synchronize the graphic signal to the score
/ITL/scene/sync myGraph score h
```

The corresponding result is given by figure 12. The signal can move and receive data in real-time. Note that the score graphic to time mapping is automatically computed by the system.

5. CONCLUSION

Our approach for synchronizing arbitrary objects in the graphic space according to their time relations, combines the advantages of simplicity and flexibility: a great variety of behaviors may be obtained depending on the defined segmentations and mappings. This method is independent of any implementation.

The proposed solution to include the performance representation into the music score is also characterized by its simplicity and flexibility. It consists in abstracting the representation computation from the rendering engine, which results in an opened and dynamically extensible system.

The resulting augmented music score supports heterogeneous components and proposes an original music notation approach, opening new spaces to music and performance representation.

There are many potential application domains, including pedagogic applications, games,... But we also hope that new music forms like interactive music, will take advantage of this research and its developments.

The Interlude Augmented Music Score framework is an open source project. The viewer is available from the Interlude web site at http://interlude.ircam.fr.

Acknowledgments
The Interlude project is funded by the Agence Nationale pour la Recherche [ANR-08-CORD-010].

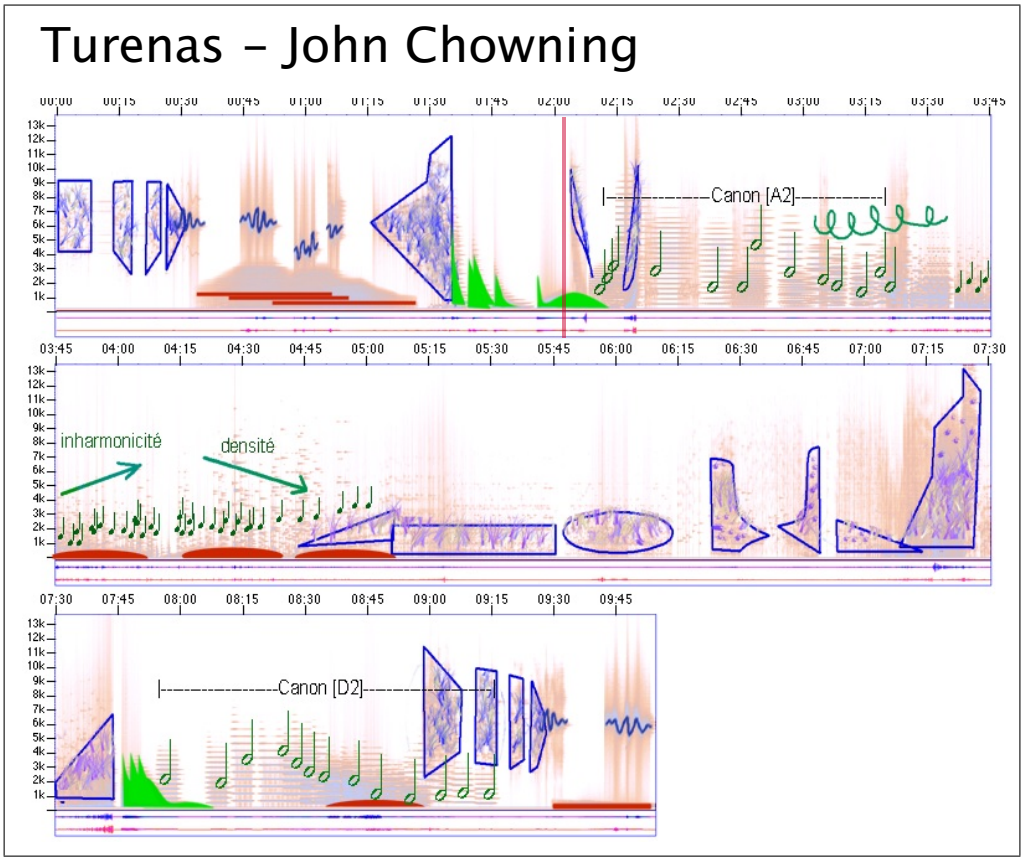


Figure 10. Turenas score: analysis and graphic transcription by Laurent Pottier. The transcription is taken from the INA-GRM "Portraits polychromes" and reimplemented using the augmented score framework.

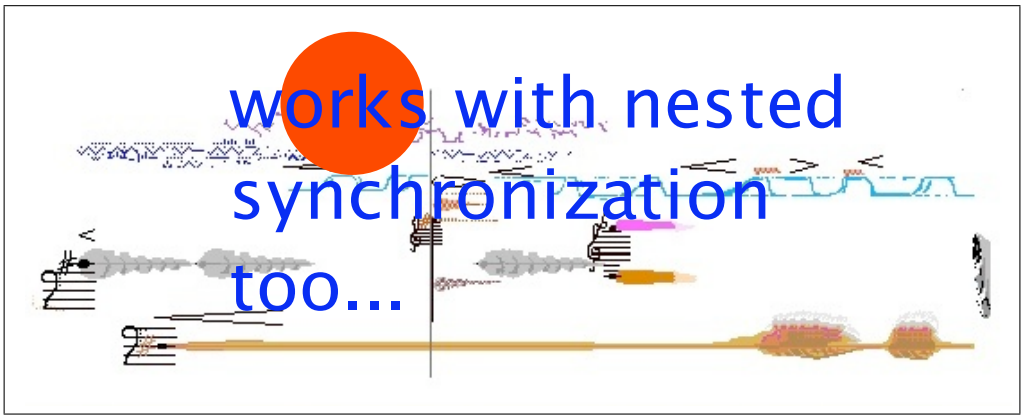


Figure 11. A score with nested synchronization. It includes a bitmap, text and a vectorial graphic. The text is synchronized to the bitmap and the circle to the text. When receiving time messages (e.g. clock), each object moves relatively to its master.

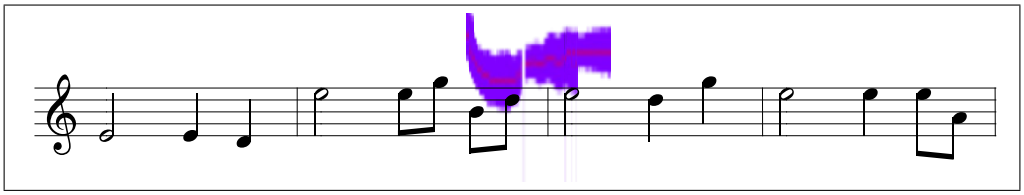


Figure 12. A graphic signal synchronized to a GMN score. The signal can move and receive data in real-time.

6. REFERENCES

[1] R. B. Dannenberg, "Music representation issues, techniques and systems," *Computer Music Journal*, vol. 17, no. 3, pp. 20–30, 1993.

[2] E. Selfridge-Field, ed., *Beyond MIDI: the handbook of musical codes*. MIT Press, 1997.

[3] W. B. Hewlett and E. Selfridge-Field, eds., *The Virtual Score; representation, retrieval and restoration*. Computing in Musicology, MIT Press, 2001.

[4] Y. Geslin and A. Lefevre, "Sound and musical representation: the acousmographe software," in *ICMC'04: Proceedings of the International Computer Music Conference*, pp. 285–289, ICMA, 2004.

[5] O. Koechlin and H. Vinet, "The acousmographe, a macinstosh software for the graphical representation of sounds," in *Proceedings of the International Computer Music Conference (ICMC'91)*, pp. 586–588, ICMA, 1991.

[6] D. Fober, S. Letz, Y. Orlarey, A. Askenfeld, K. F. Hansen, and E. Schoonderwaldt, "Imutus - an interactive music tuition system," in *Proceedings of the first Sound and Music Computing conference - SMC'04*, pp. 97–103, IRCAM, 2004.

[7] D. Fober, S. Letz, and Y. Orlarey, "Vemus - feedback and groupware technologies for music instrument learning," in *Proceedings of the 4th Sound and Music Computing Conference SMC'07 - Lefkada, Greece*, pp. 117–123, 2007.

[8] H.-W. Nienhuys and J. Nieuwenhuizen, "LilyPond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics*, 2003.

[9] M. Kuuskankare and M. Laurson, "Expressive notation package," *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[10] K. Hamel, "NoteAbility, a comprehensive music notation system.," in *Proceedings of the International Computer Music Conference.*, pp. 506–509, 1998.

[11] D. Fober, S.Letz, and Y.Orlarey, "Open source tools for music representation and notation," in *Proceedings of the first Sound and Music Computing conference - SMC'04*, pp. 91–95, IRCAM, 2004.

[12] J. Bresson and C. Agon, "Scores, programs, and time representation: The sheet object in openmusic," *Computer Music Journal*, vol. 32, no. 4, pp. 31–47, 2008.

[13] D. Baggi and G. Haus, "IEEE 1599: Music encoding and interaction," *COMPUTER*, vol. 42, pp. 84–87, March 2009.

[14] C. Cannam, C. Landone, M. Sandler, and J. P. Bello, "The sonic visualiser: A visualisation platform for semantic descriptors from musical signals," in *Proceedings of the 7th International Conference on Music Information Retrieval*, 2006.

[15] M. Wright, *Open Sound Control 1.0 Specification*, 2002.

QUANTIFYING MASKING IN MULTI-TRACK RECORDINGS

Sebastian Vega
Universitat Pompeu Fabra
svegalopez@gmail.com

Jordi Janer
Universitat Pompeu Fabra
jordi.janer@upf.edu

Third author
Affiliation3
author3@smcnetwork.org

ABSTRACT

It is known that one of the most important tasks in music post-production is equalization. Equalization can be applied in several ways, but one of the main purposes it serves is masking minimization. This is done so that the listener can appreciate the timbral qualities of all instruments within a musical mix. However, the study of masking between the different instruments of a multi-track mix has not received a lot of attention, and a quantitative measure based on perceptual studies has not yet been proposed. This paper presents such a measure, along with a study of masking between several common instruments. The measure proposed (cross-adaptive signal-to-masker ratio) is intended to serve as an analysis tool to be used by audio engineers when trying to combat masking using their preferred equalization techniques.

1. INTRODUCTION

Computers are being used to perform complex tasks that are inherently human and with a high degree of accuracy. Some examples of this are speech recognition [1] and automatic musical genre classification [2]. Recently, the possibility of a computer being able to down-mix a multi-track recording like an audio engineer is starting to be explored [3,4,5], and although the use of perceptual models has not been exploited for this purpose, it is the authors opinion that using computational models of perception might prove useful, if not indispensable, to achieve good results. The underlying complexities and non-linearities involved in the process of down-mixing are numerous, mainly because multi-track down mixing is a task where both technology and creativity co-exist in equal proportions so an exact set of rules for mixing does not really exist. In order for automatic down-mixing to become feasible many individual issues need to be tackled, for example, automatic panning, automatic dynamics control and automatic equalization. This work is a step towards the latter.

1.1 Masking within a musical context

When mixing a song, most of the decisions of an audio engineer are influenced by context; the genre of the music, the intended audience and for all we know even the weather at the time of mixing can influence the engineer. Nevertheless there is one aspect of a mix that is crucial and for which most audio engineers share the same view. This is masking. Masking has been defined as the process by which the threshold of audibility for one sound is raised by the presence of another (masking) sound. Also, as the amount by which the threshold of audibility of a sound is raised by the presence of another (masking) sound. The unit customarily used is the decibel [6].

However, when talking in terms of a musical mix, the term attains a quite different definition, namely; when one signal competes with another, reducing the hearing systems ability to fully hear the desired signal, masking has occurred [7]. The latter definition emphasizes the fact that in a multi-track mix several instruments are fighting to be heard, so the ability to fully hear every individual instrument is reduced. Accordingly, when there is a lot of masking going on between the different tracks of a multi-track recording, the resulting mix is cloudy and confusing. On the other hand, an unmasked mix is the one where all the instruments are clearly defined thus allowing the listener to fully appreciate their timbral characteristics. As such, it is the authors opinion that masking minimization should be one of the pillars of an automatic mixing system.

Luckily, masking minimization is not a mystery; it is known that audio engineers employ three weapons when combating masking: setting the relative levels of tracks appropriately (thus giving priority to some of them), panning the tracks with similar content to opposite sides of the stereo panorama, and equalization to ensure that each track is allocated a portion of the spectrum.

1.2 Motivation behind this work

It is clear that there is a lot of ground to be covered in terms of studying the factors that influence the engineers decisions when performing the three steps mentioned above. For example, if an audio engineer is using the techniques to combat masking, an analysis tool that is able to quantify masking would come very useful, this analysis tool could also be useful for automating the process of masking minimization. To the authors knowledge, a measure of masking based on perceptual studies has not yet been proposed within the context of a mix. It is then the intention of this work to provide a meaningful quantitative measure of masking between the different tracks of a multi-track recording. The proposed measure is based on the widely accepted power spectrum model of masking [8]. The next

section presents each stage of the model in detail. The interaction between several common instruments (electric guitar, bass, synthesizer) is analyzed in section 3, and conclusions are presented in section 4.

2. THE CROSS-ADAPTIVE SIGNAL-TO-MASKER RATIO (SMR)

As mentioned before the main idea behind this work was to implement an analysis tool for audio engineers to be used when equalizing a multi-track recording in the attempt to spectrally unmask the different tracks. Additionally, given that the model is able to quantify masking, another possible application of the model is in a content-based equalization system. Such an equalizer falls under the category of Adaptive Audio Effects (A-DAFX) [9]. The main idea behind this type of effect is that the processing that takes place is controlled by sound features derived from the input sound over time. Additionally, a cross-adaptive effect is defined to be a type of A-DAFX that has multiple inputs/outputs and the individual processing of a single input is dependent on the content of all the inputs. This type of processing requires a feature extraction stage (analysis stage) that takes the interaction between all the inputs into account. Furthermore, because the aim of the equalization is to minimize unwanted masking, the analysis stage should be performed on a perceptual domain and based on psychoacoustic studies. The next section introduces such an analysis stage, namely, the cross-adaptive signal-to-masker ratio.

2.1 The Power Spectrum Model of Masking

A widely accepted model of masking has been proposed by Moore [8]. The model is based on many psychoacoustical experiments that have been carried out and improved in the past few decades [6]. Most of the steps of the cross-adaptive SMR explained below have been based on this model. Briefly, to calculate the amount of masking between two sounds, the excitation patterns for the two sounds are calculated. Then, the regions with significant excitation overlap in time and frequency are detected and finally a decision is made for each of these regions in which a sound is labeled as masker and the other one as maskee (the sound that is masked). The idea is that when minimizing unwanted masking, the sounds that are considered maskers in a given region are attenuated to achieve a desired signal-to-masker ratio expressed in decibels.

2.2 Outer/Middle Ear Filtering

The first step of the model is to transform the input sounds to a perceptual representation across frequency and time. For this we must first account for the transmission of sound through the outer and middle ear. Several experiments have helped to determine the transfer function of this filter [6]; it is depicted in figure 1. The filter shows a clear peak in the region around 3 kHz and significant attenuation to low and high frequencies. The filtering is performed in the frequency domain by multiplying the magnitude spectrum of each frame of audio with the response shown in figure 1.

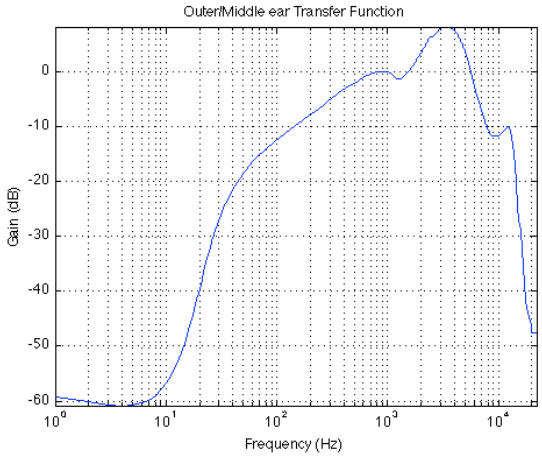


Figure 1. Magnitude Response of the outer/middle ear filter. The filtering is applied in the frequency domain by multiplying the above response with the magnitude spectrum of each frame of audio.

2.3 Excitation Pattern

The next step, after applying the outer/middle ear filtering is to calculate the excitation patterns of the analyzed sounds. The excitation pattern is meant to correspond to the average neural activity in response to a steady sound as a function of frequency [10]. It is calculated as the squared sum of the output (energy at the output) of each auditory filter described below as a function of the filters centre frequency. As such, the excitation pattern is a vector which contains the output energy of each of the 43 auditory filters of a frame of audio.

2.3.1 Auditory Filter Shape

Several experiments have been performed in order to estimate the shape of the auditory filters in the basilar membrane [8]. It has been concluded that the auditory filters take the form of a rounded exponential function. In addition it has been determined that the shape of these filters vary significantly with the level of the input sound it receives. Figure 2 illustrates this. Note that the lower frequency slope of the filter becomes less steep as the input level increases. This fact has a great implication in the case of music mixing. An engineer usually mixes a song in the 60 to 85 dB SPL range and this is also a reasonable range at which most people listen to music. This means that at this level the auditory filters in the cochlea take the corresponding form depicted in figure 2. This fact, together with the fact that the auditory filter bandwidths increase with center frequency, constitute what is known as the upward spread of masking. The upward spread of masking refers to the fact that low frequency sounds are more effective at masking higher frequency sounds. For this reason this model assumes that the music will be listened to at levels around 60-85 dB SPL, thus the auditory filters that are implemented are asymmetric, having an approximate

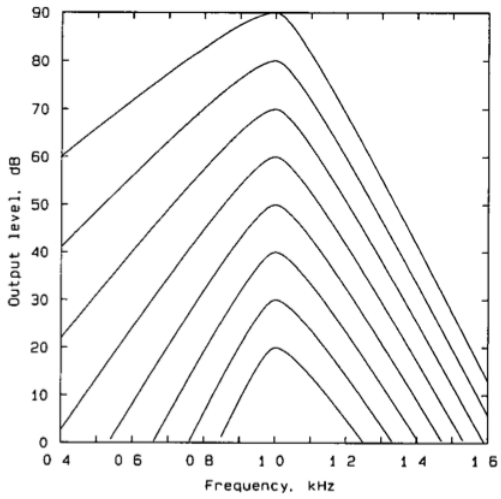


Figure 2. The response of the auditory filter has been found to change with input level. In the figure we can see the response of the auditory filter centered at 1KHz for different input levels ranging from 20 to 90 dB SPL. As you can see, the lower frequency slope becomes less steep with increasing input level.

shape to the one shown in figure 2 when the input level is 70 dB SPL.

2.3.2 Auditory Filter Bandwidths

The bandwidths of the auditory filters have also been found to change across frequency [6]. In this particular model the bandwidths are made to increase as a function of center frequency according to the Equivalent Rectangular Bandwidth scale:

$$ERB = 24.7(4.37F + 1) \quad (1)$$

Where F is frequency in kHz. This scale differs from the traditional Bark scale in that the bandwidths keep decreasing below 500 Hz [6]. Figure 3 shows the relationship between ERB band number and ERB bandwidth. In this model, the excitation pattern is calculated using 43 rounded exponential auditory filters spaced one ERB apart starting at fc = 50 Hz.

2.3.3 The Excitogram

The last step in calculating the Excitation Pattern is simple. The energy at the output of each auditory filter is calculated as a function of the filters centre frequency and the resulting vector constitutes the excitation pattern at a given time. Additionally, because we are interested in the sounds temporal evolution we calculate an excitation pattern over time to form the excitogram [10]. Two example excitograms are depicted in figure 4 and figure 5. They were calculated with a window size of 46 ms and a hop size of 23 ms. The auditory filtering was applied in the frequency domain as a set of multiplications with the filters responses.

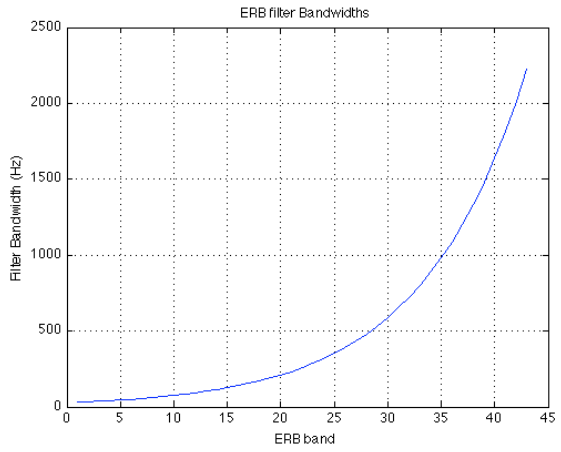


Figure 3. The relationship between the ERB band (1-43) and the auditory filter bandwidth at that band is shown in the figure.

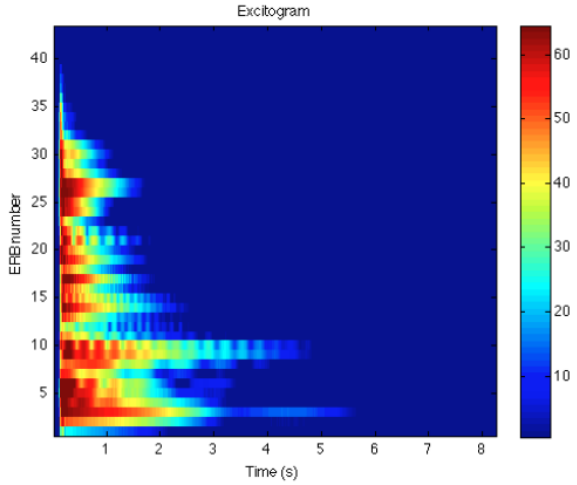


Figure 4. The excitogram of a guitar strum (Am). It can be seen that higher frequencies decay faster than lower frequencies, this is an acoustic property of vibrating strings.

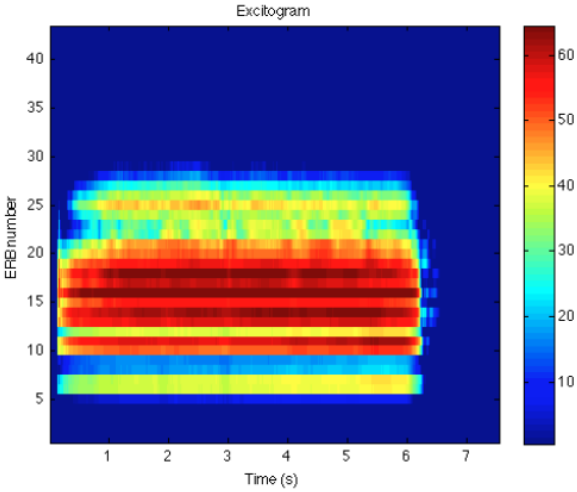


Figure 5. The excitogram of a sustained trumpet note (C3). It can be seen that this note is fairly stationary.

2.4 Masking Coefficient

The next step in the model is the calculation of the masking coefficient. This is a number between 0 and 1 that corresponds to the amount of effective excitation overlap between two sounds. The masking coefficient between two excitations is given by the following equation:

$$MC(t,b) = 1 - \left(\frac{|e_1(t,b) - e_2(t,b)|}{60dB} \right) \qquad (2)$$

Where e_1 and e_2 are the matrices containing the excitograms of each sound in decibels. The absolute value of the difference between the excitations is normalized with respect to 60 dB and then subtracted from one. As a result, when the excitograms have a similar value for a given ERB band and time, the MC is close to one. This specific case would yield a low signal-to-masker ratio in that band, as the auditory filter gets a similar amount of energy from both sounds, so this means that both sounds are heavily competing to be heard. As this is the definition of masking we have adopted (see section 1.1) we call this descriptor the masking coefficient. In practice, all of the values of the excitations below a certain threshold th_{ex} were discarded in the computation of the Masking Coefficient; this was done to avoid having high MC values at regions where the excitation values were low because it is assumed that these components are masked by the louder components anyways. Also, the values of the MC which are less than a given threshold th_{mc} are set to zero as we believe they are not significant. The values of th_{mc} as well as th_{ex} were obtained experimentally, but they could be turned into user-controlled parameters. In the example above values of th_{ex} = 20dB and th_{mc} = 0.5 were used. It is worth to analyze the MC descriptor depicted in figure 6. Basically, the MC descriptor above is saying that there is significant masking going on at the beginning of the sound (the first 5 seconds or so) and only in bands 1-28 approximately. If we look at the excitations of the sounds in the figures 4 and 5 we can see that the trumpet sound is longer than the guitar strum. This is why in figure 6 there is no masking detected after the sound of the strum has died off (around second 5). Also, as you may know, the high frequencies in a guitar strum tend to decay faster than the low frequencies; this means that the trumpet note and the guitar strum are no longer competing to be heard in the higher bands after the high frequencies of the strum have decayed. This can also be seen in figure 6.

2.5 The Cross Adaptive SMR

The last step in the model is the computation of the cross-adaptive signal-to-masker ratio. This means that a decision needs to be made in which one of the sounds is considered a masker and the other one a maskee in a given band and time. This is done because it well may be that a guitar sound can be masking (competing with) the bass in the lower bands, but it can be masked by the trumpet in the higher bands. In order to do this, the model makes use of two excitation descriptors, namely, the excitation centroid EC and the weighted average of the three most excited

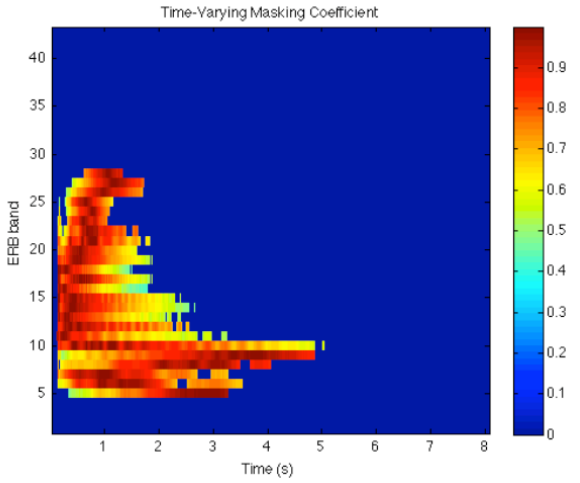


Figure 6. The masking coefficient descriptor is a measure of the effective excitation overlap between two sounds. The figure above shows the MC between the guitar strum and sustained trumpet note of the examples above.

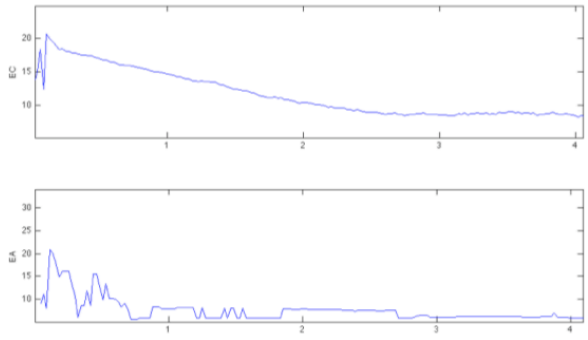


Figure 7. Descriptors EC (top) and EA (bottom) for the guitar strum.

auditory filters ERB band number (weighted by their excitation value), from now on EA. These descriptors are calculated in order to estimate the spectral region in which the sounds should live. For example, a bass should live well below a guitar and a guitar should live well below the cymbals, and these descriptors are able to give us this information. The masker/maskee estimation can be better illustrated with an example. In the case of the guitar strum sound (s=1) and trumpet note sound (s=2) both descriptors are calculated for both sounds. Figure 7 and figure 8 show the calculated descriptors for both sounds. Based on these

descriptors a decision is made like follows. For each of the time/frequency regions where the MC reveals masking the following distances are calculated:

$$\Delta c_s = |b_t - EC_{t,s}| \qquad (3)$$

$$\Delta c3_s = |b_t - EA_{t,s}| \qquad (4)$$

Where b_t is the band where masking was detected (by the MC descriptor) at a given time t. $EC_{t,s}$ is the excitation

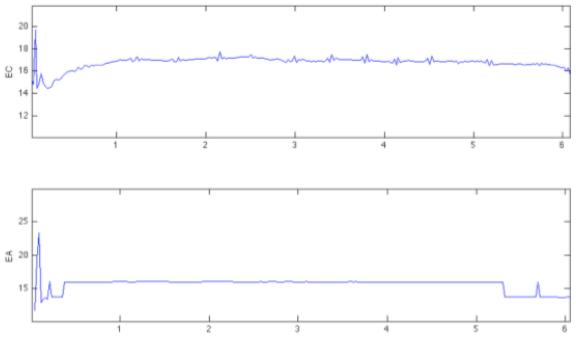


Figure 8. Descriptors EC (top) and EA (bottom) for the sustained trumpet note.

centroid of track s at time t. From these values $(\Delta c_1, \Delta c_2)$ we can figure out if the spectral region in which the sounds were competing is closer to the centroid of either sound, which we are assuming is the region around which the sound should live. Therefore, the sound responsible for the smallest value of Δ (either using equation 3 or 4) is considered to be the maskee (the sound that is being masked), and the other sound is considered the masker. It has been determined that the descriptors mentioned above behave differently depending on the sound. Sometimes the centroid is smoother and sometimes the 3-peak average ERB value is smoother. The pros and cons of using either descriptor for the analysis are still under investigation but as an initial suggestion, the descriptor that shows the smoothest evolution should be chosen. An example of such decision using only the centroid (only using equation 3) is depicted in figure 9. The red regions corresponds to the regions where the trumpet is considered a maskee, the green region corresponds to the regions where the guitar is considered a maskee. In the region where a sound is considered a maskee the sound is protected in the sense that in order to increase the signal-to-masker ratio, the other sound will be attenuated in that band. In this specific case, the sound of the guitar in the higher bands was protected at the beginning of the sound, but once the higher frequencies decayed, the guitar was assigned to the lower bands and the trumpet was assigned to the higher bands. This demonstrates the ability of the system to dynamically adapt to the relationship between the two sounds. Apart from being able to allocate a portion of the spectrum to each sound, the proposed model is also able to quantify the amount of masking between the tracks, this means that an audio engineer could keep track of exact values that work for given situations. To quantify the amount of masking we calculate the SMR which is explained next.

Finally, now that the sounds have been labeled as masker/maskee, it is possible to calculate the signal-to-masker ratio at any auditory filter at any point in time. The SMR is then given by:

$$SMR_{t,b} = 10log_{10}\left(\frac{Es_{t,b}}{Em_{t,b}}\right) \quad (5)$$

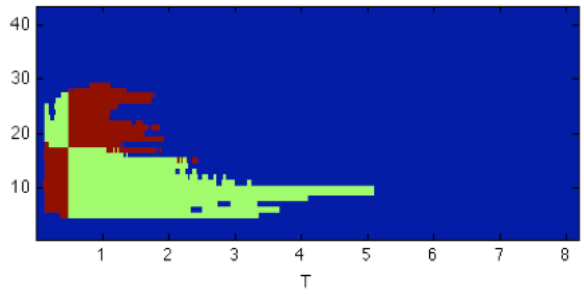


Figure 9. The masker/maskee decision for the guitar strum and sustained trumpet note. It can be seen that the decision dynamically adapts the content of both sounds. In this case, equation 3 is responsible for the decision.

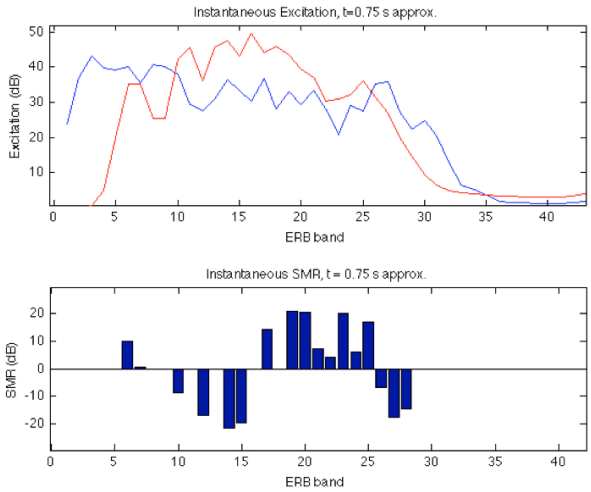


Figure 10. Instantaneous SMR for guitar strum and sustained trumpet note. The decision function is responsible for labeling the sound as masker (Em) or maskee (Es).

Where Es is the excitation of the maskee (protected signal) at time t and band b, and Em is the excitation of the masker at time t and band b. This is the measure that is used to quantify masking between the different instruments, it is expressed in decibels. An example of the SMR measure for a frame (around second 0.75) of the sounds above is depicted in figure 10. As you can see, the SMR does not have a value for some bands; this means that the MC did not detect any masking in those areas. The MC used to calculate the above SMR was calculated with a $th_{mc} = 0.8$ which means that only areas with a high degree of excitation overlapping are considered. The SMR can have both positive and negative values, a value close to 0 indicates a similar amount of energy of both sounds in that auditory filter. A positive increasing value indicates less masking (less competition) and a negative value indicates more masking (the energy of the masker is higher in that auditory filter, so there is more competition). To summarize, the whole process of calculating the cross-adaptive SMR is made up of four fundamental blocks. The first step is to calculate the excitations of the input tracks by first filtering each input track with the outer/middle ear filter, then

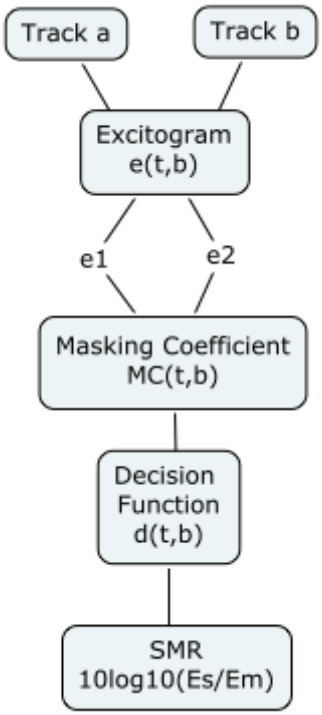


Figure 11. Overview of cross-adaptive SMR computation.

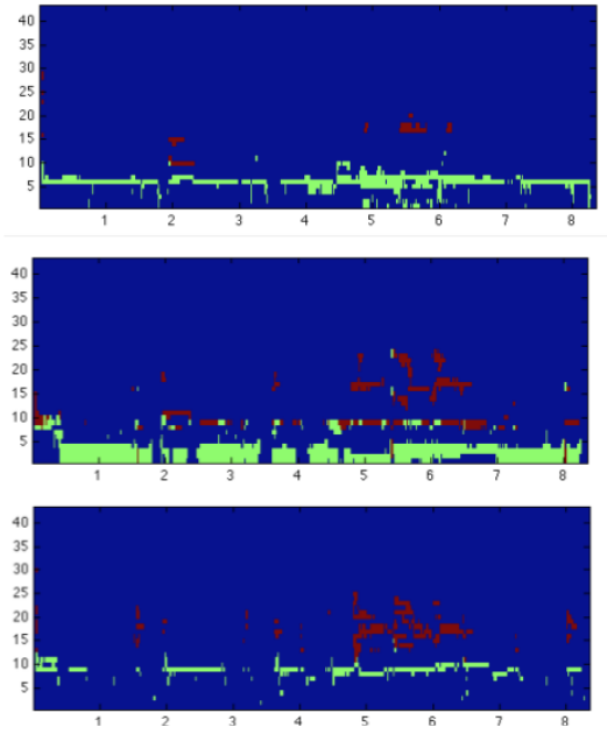


Figure 12. Top: bass and electric guitar 1. Middle: bass and synth. Bottom: bass and electric guitar 2.

passing them through an auditory filter-bank and summing the squared energy of each filter's output as a function of the filters centre frequency. Then, the masking coefficient is calculated using equation 2. Then, in order to label the tracks as either masker or maskee, both the excitation centroid (EC) and the MC descriptor are used in equation 3, or alternatively, the 3-peak excitation centroid (EA) and the MC descriptor are used in equation 4. Once the sounds are labelled across frequency and time (see figure 9) it is possible to use equation 5 to obtain the SMR. Remember that the auditory filter-bank is made up of 43 rounded exponential filters, each spaced one ERB apart (equation 1), with the first filter at $f_c = 50$ Hz.

3. ANALYSIS OF COMMON INSTRUMENTS

This section is concerned with the analysis of the proposed model. Four common instruments of a multi-track recording are selected, namely the bass, rhythm electric guitar 1 (playing chords), synthesizer, and electric guitar 2 (playing arpeggios an octave above the chords). We will see how these tracks mask each other across frequency and time in a song excerpt of length 9 seconds. The analysis is done with a graphical user interface that allows setting the relative levels of the tracks before they are analyzed. This is very important as the relative levels between tracks dictate the amount of masking between them. The tracks are set to the same relative levels for all cases (or very similar), the idea is to investigate their interaction. The first case to be explored is the bass. Figure 12 shows the result of the analysis between the bass and the rest of the tracks. The regions where the bass is masked are shown in green. As you can see, in all of the cases the bass was found to be masked in the lower bands, this agrees with intuition as it is known that the bass guitar should be the dominant instrument in the lower ranges. In this specific case, the instrument that seems to be interfering the most with the bass is the synthesizer, and the instrument that interferes the least with the bass, is the electric guitar 2. Following is the analysis of the rhythm electric guitar 1. The images below show the results of the analysis between the guitar and the rest of the tracks. The regions where the electric guitar 1 is masked are shown in green, and the regions where the electric guitar is masking another sound are shown in red. As you can observe, the electric guitar 1 is masking more bands of other sounds than the bass did. Also, the instrument that seems to mask the electric guitar 1 the most is the electric guitar 2, followed by the synthesizer. An interesting fact to observe is that the electric guitar 2 is masking the electric guitar 1 in band 10, but the synth is not (at least for most of the time). All of these subtle details can be useful to an audio engineer when equalizing the sounds.

Next we will observe the interaction between the synthesizer and the rest of the sounds. Figure 13 shows the results of the analysis between the synth and the rest of the sounds. In the case of the synth we can observe that the only instrument that is masked by the synth in the lower bands is the bass. Also, the electric guitars both show a similar interaction with the synth, although electric guitar 1 seems to be interfering with the synth in lower bands. From the visualization of these results it is possible to make some conclusions once the interaction between all tracks has been analyzed. First of all, the bass track was

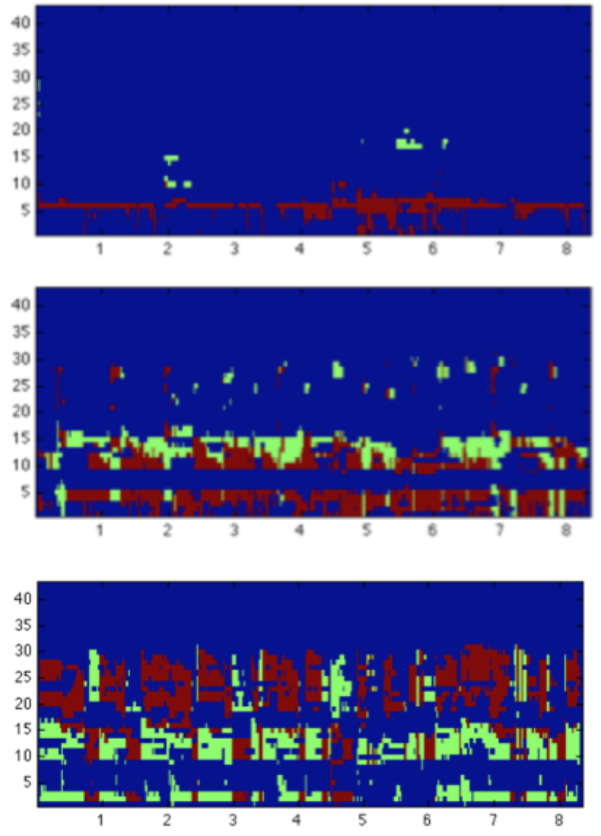


Figure 13. Top: electric guitar1 and bass. Middle: electric guitar1 and synth. Bottom: electric guitar1 and electric guitar 2.

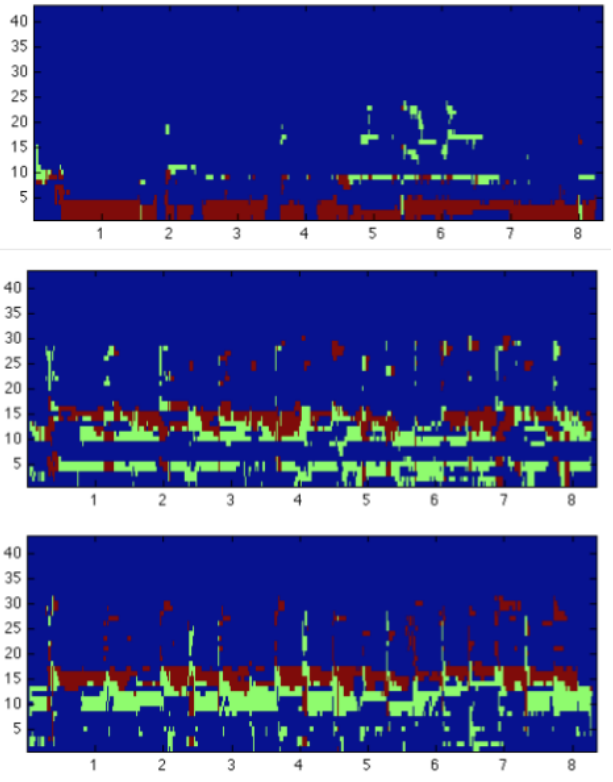


Figure 14. Top: synth and bass. Middle: synth and electric guitar1. Bottom: synth and electric guitar 2.

determined to be masked in the lower bands for all cases, this means that this instrument should definitively occupy the lower bands and the rest of the tracks should be attenuated at those bands to increase the signal-to-masker ratio of the bass in a given region. Secondly, The rhythm electric guitar 1 was masked below band 5 and above band 10 by the electric guitar 2 and it was masked only above band 10 by the synth, this implies that the electric guitar 2 should live higher in frequency than the electric guitar 1, which in turn should live higher in frequency than the synthesizer.

4. CONCLUSIONS

A quantitative measure of masking between the different tracks of a multi-track recording has been implemented. We have seen that the model is able to decide if an instrument is acting as a masker or as a maskee in a time/frequency representation by means of analyzing its spectral content. This decision has been analyzed with several common instrument tracks and we have shown that the model is able to allocate a portion of the spectrum to each instrument, which is a task that most audio engineers need to develop a skill for. Furthermore, once the decision has been made we can calculate the SMR for any band at any point in time where masking is detected which means that masking can be quantified. Also, the visualization of this decision can be very useful for audio engineers when equalizing tracks to reduce masking, which is known to be an important task in music post-production [7]. Furthermore, the proposed model could also be used as the analysis stage of an adaptive audio effect in the future that could automate the masking minimization process, although this matter is still under investigation.

5. REFERENCES

[1] P. Green, M. Cook, *Robust automatic speech recognition with missing and unreliable acoustic data*. Speech Communication, Vol 34, Issue 3, pp-267-285 June 2003.

[2] G. Tzanetakis, P. Cook *Musical genre classification of audio signals*. IEEE transactions on speech and audio processing, Vol 10, No. 5, July 2002.

[3] E. Perez, J. Reiss *Automatic equalization of multichannel audio using cross-adaptive methods*. Audio Engineering Society, 127th Convention paper, October 2009.

[4] E. Perez, J. Reiss *Improved control for selective minimization of masking using inter-channel dependency effects*. Proceedings of 11th DAFx International Conference, September, 2008.

[5] E. Perez, J. Reiss *Automatic mixing: live down mixing stereo panner*. Proceedings of DAFx International Conference, Bordeaux, France 2007.

[6] Brian Moore, *An introduction to the psychology of hearing*. Academic Press, 5th Edition, ISBN-13: 978-0125056281, 2003.

[7] A. U. Case *Sound FX: unlocking the creative potential of recording studio effects*. Focal Press, ISBN-13: 978-0-240-52032-2, 2007.

[8] Brian Moore *Masking in the human auditory system*. Collected papers on digital audio bit reduction, March 1995.

[9] V. Verfaille, U. Zolzer *Adaptive digital audio effects (A-DAFX): a new class of sound transformations*. IEEE transactions on audio speech and language processing, Vol 14, No. 5, September 2006.

[10] J. Smith, R. Cassidy *Psychoacoustics lab*. Real Simple Project, CCRMA, Stanford University.

[11] A. Seefeldt, *Loudness domain signal processing*. Audio Engineering Society, 123rd Convention paper, No. 7180, October, 2007

[12] M. Hoffman, P. R. Cook, *Feature-Based Synthesis: Mapping Acoustic and Perceptual Features onto Synthesis Parameters*. Department of Computer Science, Princeton University.

[13] B. Moore, B. Glasberg, T. Baer, *A model for the prediction of thresholds, loudness and partial loudness*. J. Audio Eng. Soc. Vol 45, No. 4, April 1997

[14] E. Perez, J. Reiss, *An automatic maximum gain normalization technique with applications to audio mixing*. Audio Engineering Society 124th Convention, May 2008

[15] E. Pampalk, A. Rauber, D. Merkl, *Content-based organization and visualization of music archives*. ACM, 2002, pg 570-579.

[16] X. Amatriain, J. Bonada, A. Loscos, J. Arcos, V. Verfaille, *Content-based transformations*. Journal of New Music Research, Vol 32, No. 1, pp. 95-114, 2003.

[17] V. Verfaille, D. Arfib, *Implementation strategies for adaptive digital audio effects*. Proc. of the 5th International Conference on Digital Audio Effects, DAFx, September 2002.

[18] J. Bitzer, J. LeBoeuf, U. Simmer, *Evaluating perception of salient frequencies: do mixing engineers hear the same thing?*. Audio Engineering Society, 124th Convention paper No. 7462, May 2008

[19] E. Skovenborg, S. Nielsen, *Evaluation of different loudness models with music and speech material*. Audio Engineering Society, 117th Convention paper, October 2004.

[20] D. Reed, *A perceptual assistant to do sound equalization*. IUI, New Orleans, ACM, 2000.

[21] J. Bitzer, J. LeBoeuf *Automatic detection of salient frequencies*. Audio Engineering Society, 126th Convention paper No. 7704, May 2009

[22] E. Lopez-Poveda, R. Meddis, *A human non-linear cochlear filter-bank*. Journal Acoust. Soc. Am 110 (6) December 2001

MIXTRACT: AN ENVIRONMENT FOR DESIGNING MUSICAL PHRASE EXPRESSION

Mitsuyo Hashida **Shunji Tanaka** **Takashi Baba** **Haruhiro Katayose**
Research Center for Human & Media,
Graduation School of Informatics, Kwansei Gakuin University, JAPAN
{hashida, s.tanaka, takashi-b, katayose}@kwansei.ac.jp

ABSTRACT

Music performance is processing to embody musical ideas in concrete sound, giving expression to tempo and dynamics and articulation to each note. Human competence in music performance rendering is enhanced and fostered by supplementing a lack of performance skill and musical knowledge using computers. This paper introduces a performance design environment called Mixtract, which assists users in designing "phrasing," and a performance design guideline called the Hoshina-Mixtract method executable on Mixtract. Mixtract provides its users with a function for assisting in the analysis of phrase structure and a function to show the degree of importance of each note in a phrase group. We verified that the proposed system and method help seven children to externalize their musical thought and help them transform their subjective musical thoughts into objective ones.

1. INTRODUCTION

Making and performing music is regarded as very effective ways to foster musical competence. However, in many cases, lack of control or skill with musical instruments can limit the ability to test a personal free musical thought.

If people can use a computer system to test their musical thought, listen to their attempts in the form of sound feedback, and receive assistance with their limited skill, their musical competence will likely increase.

Some music composition systems that transform graphic materials into sound are available for this goal. UPIC [2], designed by Iannis Xenakis in 1977, is the most famous and historically important system. The software version of UPIC, called Iannix, is currently available in a Max/MSP environment. For a commercial software package, Kid Pix, released in 1989 [7] for Macintosh, is regarded as the first edutainment application for children to enjoy creating sound sequences by drawing graphic icons. One study used interactive composing environments to return a musical passage in response to a player's input, which has been used for music education purposes [3].

This study deals with competence in performance rendering rather than musical composition. Performance rendering is a process of creating a musical idea as a vivid movement of loudness, tempo and articulation of note sequences. Musical performance interfaces based on beat tapping or conducting interfaces are available for this purpose [9][11][12]. In contrast with these real time applications, our study aims at letting users more carefully consider performance rendering based on phrasing. Phrasing is musical expression related to vocalization and respiration. This musically important expression is easy for people to grasp. However, it is not easy to understand how to control phrasing. Moreover, structuring phrasing is difficult without guidelines. The authors have been developing a performance design system called Mixtract to cope with this difficulty and have formalized a music interpretation theory to be used on Mixtract [6]. That is, we provided an environment to test musical thought regarding phrasing with sound feedback for users who lack performance skills.

The construction of this paper is as follows. In the next section, phrasing and an overview of Mixtract are introduced. The Mixtract overview is followed by explanations of each function for assisting in phrasing design. Then, practical phrasing design methods using Mixtract and the preliminary evaluation executed for elementary and junior high school students are described.

2. OVERVIEW OF MIXTRACT

2.1 Phrasing and Mixtract

A common property of all music is that all adjacent notes are musically connected, and the sequence of successive notes forms a musically meaningful group, which is referred to as a "phrase." Phrases are also combined with each other to construct the phrase structure.

The phrase structure of a musical piece is not always unique. There are many possible candidate phrase structures. Musicians and conductors select one of the possible phrase structures and embody it in sound, as the structure is conveyed to the audience, as shown in Figure 1. There are various ways to express a phrase structure. The com-

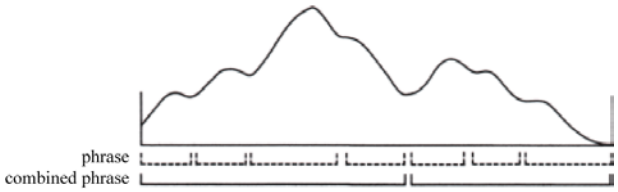


Figure 1. An image curve of phrasing (from [8]). The horizontal axis shows the timeline, and the vertical axis shows dynamics and tempo. Phrases consist of a hierarchical structure, and there are rainbow-shape curves of dynamics, tempo and articulation in each phrase.

bined expression of tempo, dynamics and the articulation of each note of the piece are heard as the individuality of the performer.

In the construction of a design assistant system, it is important to ensure considerable flexibility for users. However, too much flexibility can confuse users. The following two priorities must be considered in designing a musical design assistant system: (1) The system can function as an environment in which to test a user's musical ideas with real sound, and (2) the system provides automatic functions to complement the user's input while maintaining the user's intention. Mixtract is a musical expression design-supporting environment based on the above ideas.

Figure 2 shows an overview of Mixtract. Users of Mixtract first put a score into the system by hand on the piano roll editor view or by importing a MusicXML file. Next, they specify groups of notes as phrases and give them expression (dynamics, tempo and articulation) curves. Musical expressions given by expression curves are consistently consolidated into the whole expression. The onset time, the offset time and the dynamics of each note may be individually edited, if desired. Every time the user edits one of the expression curves, the piano roll that shows the performance preview is updated. Mixtract allows users to listen to the performance at any editing stage. This feedback is effectively used in performance design.

Our policy in designing Mixtract is that users are responsible for performance design tasks, including determination of phrase structure and editing the expression curves. Automatic processing technology is employed to fill in where users lack skill or have no special requirements. The following functions based on cognitive music theories are provided for this goal.

2.1.1 Hierarchical Phrase Structure Analysis

In order to free users from the tedious work of giving a hierarchical phrase structure to the system, Mixtract supports an automatic analysis of hierarchical phrase structure that preserves user-provided phrases. This function is based on *A Generative Theory of Tonal Music* (GTTM) [10].

GTTM itself does not offer a solution to ambiguous phrase structure. It can, however, effectively make up the remainder of the structure that the user does not directly specify. The detailed design of the interactive features of GTTM will be described in a later section.

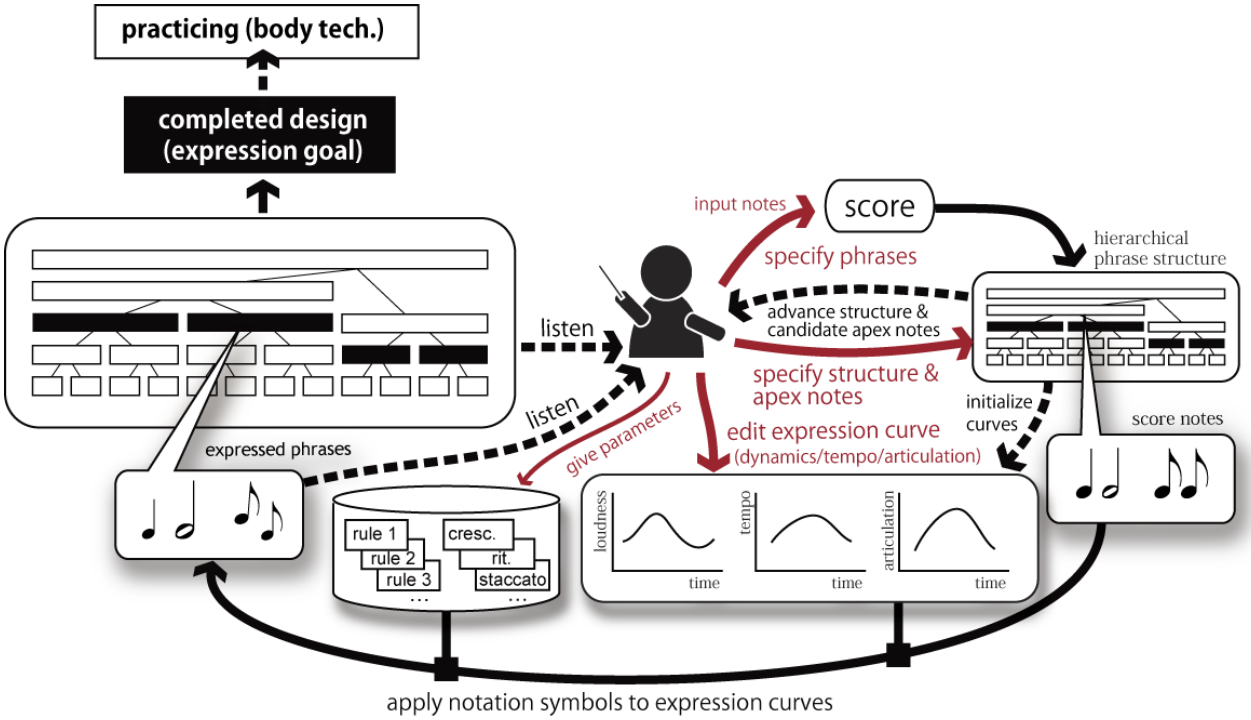


Figure 2. Overview of Mixtract

2.1.2 Phrasing Design Based on Apex Note

There are two well-known principles of phrase expression: (a) giving an accent of velocity to the beginning note of a group and (b) rainbow-shape expression [1]. Rainbow-shape expression increases dynamics and tempi from the beginning note and then decreases them to the last note of that group.

These principles of phrase expression are simple. However, it is confusing to decide whether to employ (a) or (b) and to find the apex note of phrase in (b). To solve this difficulty, we formalized a musical interpretation theory [8] proposed by Hoshina, one of the best supervisors of directing brass bands in Japan. He showed the existence of the apex notes of phrases and an outline of how to find the apex notes through score analysis. Mixtract provides users with an apex probability viewer, as shown using several shades of red in Figure 3. A policy of editing an expression curve to make the apex fall at the apex note contributes to reducing the amount of trial and error in finding good expression curves.

3. EDITING PERFORMANCE CURVES AND SYNTHESIS OF EXPRESSION

Mixtract provides a performance design GUI using expression curves for each phrase and an editor for the start timing, duration and dynamics of each note. In this section, we illustrate the performance design interface using expression curves and describe its representation algorithm.

3.1 Editing Expression curves

When a phrase structure is fixed, the system prepares default shapes for the expression curves: tempo curve, dynamics curve, and articulation curve. When users of the system click a phrase, the expression curve editor, as shown in Figure 3, appears. The users participate the performance design by editing the expression curves.

3.2 Tempo Curve and Calculation of Note-on and Note-off Timing

A tempo curve value is given using exponent representation; 0: no change, 1: tempo-up to double, -1: tempo-down to half. The total tempo expression of hierarchical phrases is expressed by the following equation:

$$Tempo(t) = \sum_{k=1}^{n} \omega_k \times GroupTempo_k(t) \qquad (1)$$

where $GroupTempo_k(t)$ is the tempo curve of the phrase that locates the score time t, and W_k is the weight of the curve of the phrase. Note-on and note-off timing are calculated by the integration of $Tempo(t)$. The real time between score time t_0 and t_1, denoted as $Time(t_0, t_1)$, is calculated using the following equation:

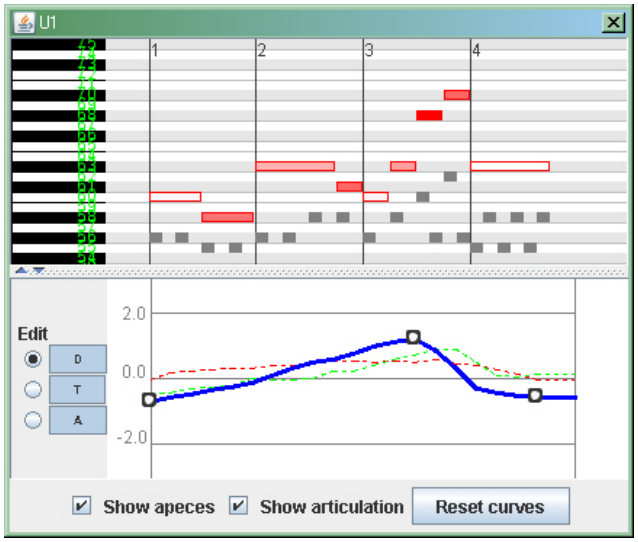


Figure 3. Performance curves and the editor with the apex probability viewer. The toggle buttons for dynamics (blue), tempo (green) and articulation (red) turn each expression curve on or off. The curve selected by the left radio button can be edited using a mouse. The apex probability of notes is shown using shades of red in the apex probability viewer.

$$Time(t_0, t_1) = \sum_{t=t_0}^{t=t_1} DeltaDuration \times 2^{-Tempo(t)} \qquad (2)$$

where *DeltaDuration* is the value prescribed by the average tempo (beats per minute) and resolution of the integration range. This equation makes the expression of the gradual tempo change within a beat portable to a real time scheduling application such as the conducting interface.

3.3 Dynamics curve and calculation of velocity

The overall dynamics of the note is given by the summation of the dynamics curves of the hierarchical phrases, as in the timing calculation.

$$Dyn(t) = \sum_{k=1}^{n} \omega_k \times GroupDyn_k(t) \qquad (3)$$

The dynamics of the note at score tine t is given by the following equation, as the dynamics is calculated with velocity of MIDI expression.

$$Vel(t) = StdVel + Dyn(t) \qquad (4)$$

where *StdVel* is the default standard value (64).

3.4 Articulation Curve and Calculation of Note-off Timing

Articulation refers to a performance technique, which affects the transition or continuity on single note to the following note. The current version of Mixtract deals with digital keyboard instruments. In this situation, the

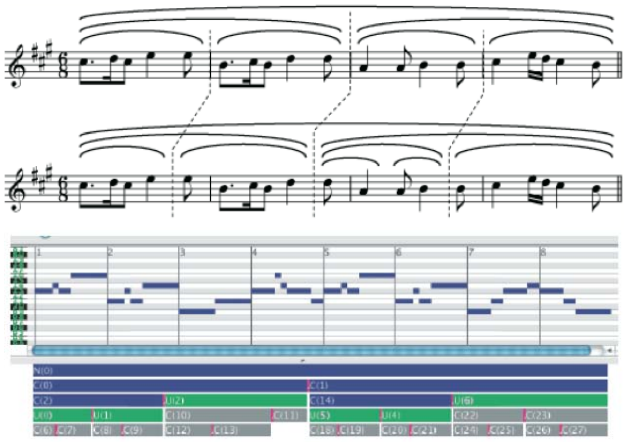


Figure 4. Phrase structure. Green boxes are user-specified phrases, while blue and gray boxes are the system-analyzed hierarchical phrases based on the user's phrases. Red lines are phrase boundary candidates estimated by the system.

major control parameter for articulation is the timing of each note-off.

The score time of the *i*th note offset, denoted as $N_{offset}(i)$, is revised using the following equation:

$$N_{Offset}(i) = N_{Onset}(i) + N_{TV}(i) \times GroupArtcltn(N_{Onset}(i) + N_{TV}(i)) \quad (5)$$

where $N_{Onset}(i)$ and $N_{TV}(i)$ are score onset time and time value of the *i*th note respectively. $GroupArtcltn(t)$ is the ratio of the articulation curve of the phrases at score time *t*. The real time of $N_{Offset}(i)$ to be issued is calculated using equation (2).

3.5 Expression of Expression Marks

For the expression marks such as crescendo and staccato, which are explicitly described in the score, Mixtract provides a replacement function in the form of expression curves. Like the expression curves of phrases, these curves can be edited.

3.6 Chords and Polyphony

The data description in the current version of Mixtract adopts procedures to place all notes in a single voice part, namely, single staff notation. It is possible to generate performance expression in melodies consisting of multiple voice parts, including the chord; however, the current system is not intended for the performance design of polyphony and tempi, each part of which may proceed differently. System implementation for polyphony is one of our future plans.

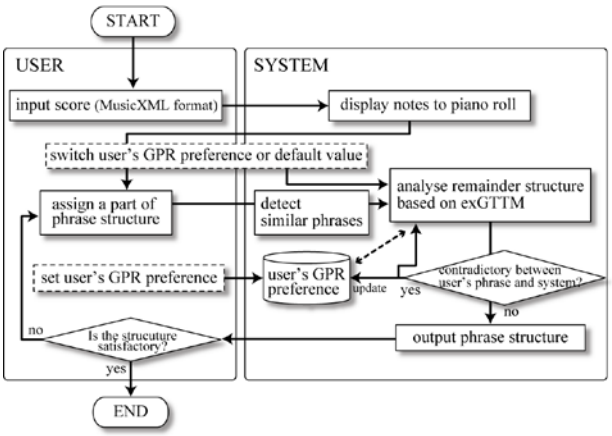


Figure 5. Outline of phrase structure analysis

4. HIERARCHICAL PHRASE STRUCTURE ANALYSIS FUNCTION

Musical phrase structure is ambiguous. Mozart's Piano Sonata K.331 (see Figure 4, upper part) is often cited by books dealing with musical theories as an example of this ambiguity. Mixtract provides a function that automatically analyzes the remainder of the phrase structure that the users do not assign. The users give the primary phrase boundary that they especially want to assign when listening to melodies. In the lower part of Figure 4, the phrases in green are those given by the user, and the phrases in other colors are phrase structure obtained automatically.

The goal of the design of the structure analysis is to reflect the user's intention and direct manipulation. Figure 5 shows the overview of the function for phrase structure analysis. If the phrase part analyzed automatically differs from what the user wants, the user can edit the phrase boundary position again, and the system will analyze the remainder again. This procedure is repeated until the user obtains the desired phrase structure.

4.1 Reflection of User's Intention

The function for automatic analysis of phrase structure is based on exGTTM [5]. ExGTTM is an extension of GTTM [10], because it may work as a computational model.

There are likely to be cases in which the users' direction contradicts the automatic analysis. Mixtract gives priority to the user's grouping direction. The lower and the upper structures of the concerned phrase are combined or divided, preserving the users' direction. Then, each default preference weight of the contradictory rules for grouping of exGTTM is revised to reflect the user's preference. The user is allowed to choose the default preference weight or the revised preference weight the next time the automatic phrase structure analysis is applied. The system also supports a function to find the same motives that the user assigned, which also helps to reduce the tediousness of the task.

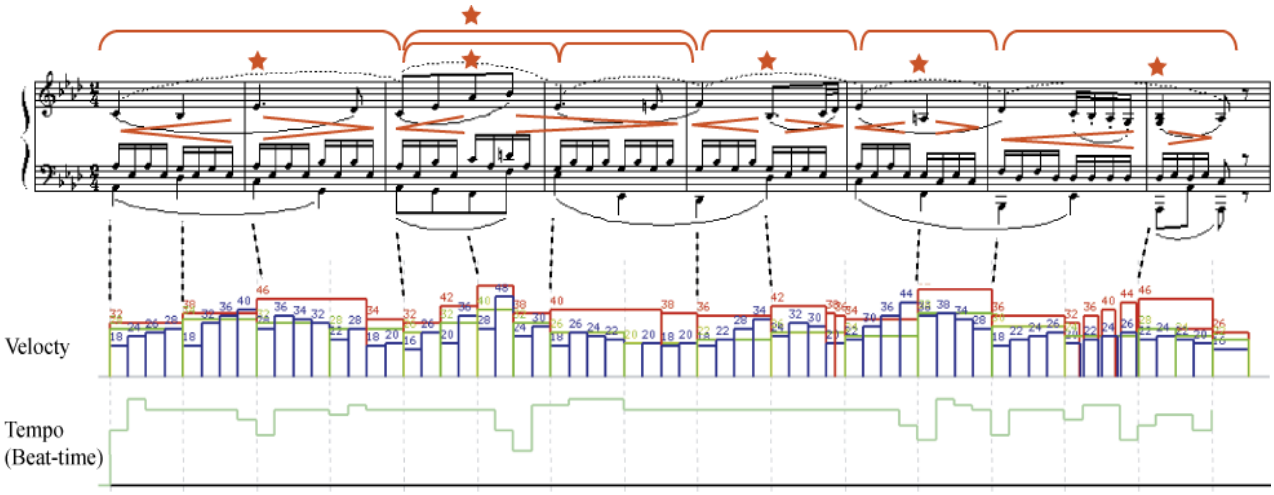


Figure 6. Hoshina's interpretation and Expression of the 2nd Movement of Beethoven's Piano Sonata "Pathetique": The annotated slurs are according to the Henle Edition. The brackets above the main melody indicate Hoshina's grouping, and the stars indicate the apices of those groups. The Crescendo and diminuendo are written by the author based on his expression.

4.2 Direct Manipulation

When users of exGTTM want to change some phrase boundaries, they have to influence the system's behavior by specifying the preference weight of each rule, Instead of this indirect operation; Mixtract users are allowed to edit the phrase boundaries directly. They can assign phrases intuitively by just listening to the music, without any knowledge of the meaning of each rule for grouping. This is one of the key features that nonprofessional people can use this system.

5. PHRASE APEX ANALYSIS FUNCTION

Mixtract provides a function to guide the apex note of a phrase based on the conditions that Hoshina described for how a note may be recognized as the apex: the crest in the outline of a note sequence, notes with a long time value, more strained notes in the tension-resolution structure, and so on. He also suggested that the last note of a phrase cannot be the apex. Figure 6 shows an example of apex analysis by Hoshina.

Hoshina's theory is distinctive compared with other music theories. However, it is too subjective to be directly used as a computational model. Therefore, we formalized the theory as rules, added some additional principles, and fixed the value of the point of each rule empirically as shown in Table 1 (last page). The likelihood of each note's being the apex is calculated by adding the points of the corresponding rules for the note. The result of the calculation is converted into shades of red on the piano roll, as shown in Figure 3.

6. PERFORMANCE DESIGN METHOD USING MIXTRACT AND EVALUATION

6.1 Performance Design Method

Mixtract itself is a general environment for phrasing expression design. Inexperienced musicians, including beginners and children, may be confused as to which of a lower or higher phrase should be selected to give expression or how to edit a free-hand expression curve. To reduce such confusion, we set up a set of guidelines for phrasing called HMM (Hoshina-Mixtract Method) that can be used on Mixtract, based on Hoshina's theory.

In using HMM, first the user has to specify the primary phrase line, which is a sequence of phrases composed of a few measure lengths. Some people may think that it is not easy to judge phrase boundaries simply by looking at a visual score (piano roll). However, some of the breath points of singing melodies will be phrase boundaries. Using the playback function of Mixtract, the user can specify the phrase boundaries almost intuitively. Next, the user edits each expression curve for the dynamics, tempo and articulation of the phrases of the primary phrase line, as referring the information of each phrase apex. The user directly modifies the onset time, the offset time and the dynamics of each note, if needed. The concrete steps of HMM are as follows:

Hoshina-Mixtract Method

Step 1: Specification of the primary phrase line

1. Input initial primary phrase line
2. Modify and fix the line using playback function

Step 2: Specification of the apex of each phrase of the primary phrase line

1. System shows apex candidates with probability
2. (User fixes apex notes)

Step 3: Expression design of the primary phrase line with apex information

1. System gives default parameters of expression curves based on Hoshina's theory
2. Edit expression curves
3. (Converting Expression Marks into expression curves)

Step 4: Expression design of the higher-level phrases of the primary phrase line

1. (System gives default parameters of expression curves that as they may express the phrase as a group)
2. Adjust the position of the apex and the amplitude range of each curve, if necessary.

Step 5: (Modification of the onset time, the offset time and dynamics of each note)

The procedures in parentheses are optional. The user may go back to a previous procedure at any point, if desired.

6.2 Workshops for Phrase Expression Design with HMM

We conducted a workshop to investigate the effectiveness of Mixtract and HMM in helping around children to learn phrase design. The workshop participants were five elementary school students, ten to eleven years old, and two fourteen-year-old junior high school students belonging to a brass band club at their school. The workshop was led by a high school music teacher who also teaches children to play the piano and who majored in piano and completed a master's degree in music. None of the participants had ever studied music in depth, although some could play piano and some could play a brass instrument. Figure 7 shows some photographs of the workshop.

Figure 7. Photograph of the workshop on phrase expression using Mixtract for seven children.

After the workshop, all of the participants commented, "It was interesting to listen to expression is changed by modifying depth of expression curves." One of the junior high school students commented, "I want to bring back the system to my home to think of phrase expression more deeply for my club activity." The following comment by the lecturer was more meaningful: "I reminded myself that I have never taught students phrasing as a general concept and a concrete methodology so far. I was made to reconsider how to teach music through this workshop. In this sense, I was a student in this workshop, too."

7. DISCUSSION AND FUTURE WORK

7.1 Mixtract as Phrase Design Environment

Mixtract is a music performance design framework that facilitates phrase expression. It includes functions that perform hierarchical phrase structure analysis and apex note analysis, and it provides an expression curve editor for each phrase.

The educational goal of Mixtract is to let its users externalize and formalize their tacit knowledge about musical performance. For this goal, it is crucial to give the users feedback on how the users' actions affect the result. The most recent version of the commercial notation software Finale [4] includes functions that render expression marks into expressive sound, and the performance rendering system SuperConductor [13] generates expressive performance based on the expression of metric structure. Commercial music sequencers are equipped with piano-roll visualization, and each has its own GUI. These systems contribute to improving efficiency in music performance production, but they are not necessarily good frameworks for letting the users consider phrasing.

The design of phrase expression in Mixtract is executed as non-real-time processing through the repetition of editing. Another computer-assisted approach to letting the user think about phrase expression is the use of performance systems that work with beat tapping or conducting interfaces such as RadioBaton [11], iFP [9] and FamilyEnsemble [12]. One of the biggest advantages of using these systems is that the player can express tempi and dynamics with simple physical movement, without worrying about pursuing the correct notes of scores. Real-time control and feedback are among the advantages of this approach. On the other hand, this function can be a disadvantage if the goal is to let the users formalize their tacit knowledge about phrase expression.

We believe that combining the use of Mixtract and a performance interface based on beat tapping could contribute greatly to increasing competence in musical performance, especially for individuals who lack performance skill and musical knowledge.

7.2 Future Work

As described in the implementation section, the current version of Mixtract is applicable to clearly multi-part music, but not to truly polyphonic music. Implementing a version for polyphonic music is a primary technical goal. The second technical goal is to provide an automatic harmonic analysis function to assist in apex probability analysis. This function will also be necessary when dealing with polyphony.

Experienced musicians already understand how to capture and express phrases, but this understanding is in the form of tacit knowledge, and even experienced musicians cannot always explain how expression is produced as a combination of dynamic, tempo and articulation of notes. We expect Mixtract and HMM to help externalize such tacit knowledge. In particular, workshops where participants can explore their expression are expected to raise not only their musical competence but also their musical culture level. Our workshop was only preliminary. We plan to continue workshops and to modify the method, and we plan to construct a curriculum for practical workshops executable by educational facilities in the near future.

8. CONCLUSION

We developed Mixtract as an environment to support musical performance design focusing on phrase expression. In this paper, we described the concept of the system design, the hierarchical phrase structure analysis and the phrase apex analysis, and we then showed an approach for phrasing called HMM that can be used on Mixtract, based on Hoshina's theory. We also described a preliminary evaluative study of Mixtract and HMM's performance for use in designing phrase expression by elementary and junior high school children.

Mixtract itself is a framework for performance design. Unlike most automatic performance rendering systems to date, Mixtract assists its user's music interpretation and helps to convey the musical interpretive intent to the system. In the preliminary study using HMM, we observed that children continued to edit curves and listen to the changes until they obtained the desired result. They were able to see the form of the expression curve together with the apex-likelihood. We are convinced that the children's subjective idea of phrase expression is shifted to an objective one through their own active editing and sound and visual feedback.

We are still at an early stage of providing a musical environment in which people are able to practice musical performance, especially focusing on phrasing, in an active environment. We hope to continue to perform experiments in the use of Mixtract by both of experienced and inexperienced musicians to foster musical competence in performance rendering and will analyze the results in future work.

9. REFERENCES

1. Blum, D. *Casals and the art of interpretation*, University of California Press, 1980.
2. Coduys, T., and Ferry, G. Iannix aesthetical/symbolic visualizations for hypermedia composition. In *Proceedings of Intl. Conf. Sound and Music Computing (SMC)* (Paris, October 2004), P18.
3. Ferrari, L. Addessi, A.R. and Pachet, F. New technologies for new music education: The Continuator in a classroom setting, in *Proceedings of ICMPC06* (Bologna, Italy, August 2006).
4. Finale by MakeMusic Inc. Available at http://www.finalemusic.com/
5. Hamanaka, M., Hirata, K., and Tojo, S. Implementing "a Generative Theory of Tonal Music". *Journal of New Music Research*, 35 (December 2006), 4, 249-277.
6. Hashida, M., and Katayose, H. Mixtract: A Directable Musical Expression System, In *Proc. of Affective Computing and Intelligent Interaction* (Amsterdam, September 2009).
7. Hickman, C. Kid Pix: The Early Years. Available at http://pixelpoppin.com/kidpix/
8. Hoshina, H. *An Approach to Vivid Musical Phrase Expression*, Ongaku-no-tomo-sha Corp., 1998. (written in Japanese)
9. Katayose H., Okudaira K. iFP A Music Interface Using an Expressive Performance Template, *Entertainment Computi*ng, *Lecture Notes in Computer Science, Vol. 3166, Springer* (September 2004), 529-540.
10. Lerdahl, F. and Jackendoff, R. *A Generative Theory of Tonal Music*, MIT Press, 1983.
11. Mathews, M. The Conductor Program and Mechanical Baton, In *Proc. of Intl. Computer Music Conf.* (ICMC) (Ohio State University, USA, 1989), pp.58-70.
12. Oshima, C., Nishimoto, K., and Suzuki, M. Family Ensemble: A Collaborative Musical Edutainment System for Children and Parents, *In Proc. of MM '04 and Co-Located Workshops* (New York, October 2004), ACM Press, 556-563.
13. SuperConductor by Clynes, M., and Microsound Intl. LTD. Available at http://www.microsoundmusic.com/

Category	Rule		Score point of likelihood of apex		
	target note		preceding	target	following
duration	first note in adjacent 2 notes	1) shorter than following note		0	1
		2) longer than following note		1	0
	first note of repeated notes in equivalent duration			1	
pitch	first note in adjacent 2 notes	1) lower than following note		0	1
		2) higher than following note		1	0
	second and later notes of repeated notes in equivalent duration		order number/note counts		
	progress note to reach the top (3rd note of 4 adjacent notes)	1) ↗ ↗ ↘	0	1	0
		2) ↘ ↗ ↘	2	1	0
		3) ↗ ↘ ↗	1	2	1
		4) ↘ ↘ ↗	0	2	1
		5) ave. duration is shorter than 0.25 ms	1	0	0
melodic tension-relaxation	appoggiatura	1) duration is longer than following note		3	
		2) duration is as same as following note		2	
		3) duration is shorter than following note		1	
		4) following note inverts		2	
		5) in conjunct motion		1	
		6) leap to following note		2	
		7) conjunct to following note		1	
		8) preceding note is a rest		1	
	suspension	if tied from preceding note	3	0	
		else	2	1	
	repeated note	if tied from preceding note	2	0	
		else	1	0	
	auxiliary note	ascending then descending		2	
		descending then ascending		1	
	passing note	ascending		2	
		descending		1	
	other ornaments (accidentals)			1	
on harmony	chord	I		0	
		I6		1	
		I46		3	
		II		6	
		IV		1	
		V		2	
		V7		3	
		VI		1	
	substitute chord	accidental		1	
		ordinary		2	
	chord change			1	
	cadence (dominant motion)	V(7) - I		4	0
		V(7) - I6		2	3
		V(7) - VI		2	1
	register	widen		1	
		narrow		-1	
	density	double stop		1	
as phrase	beginning note			1	
	ending note			-1	
	longest note	if including bounding division		1	
		else		2	
	highest note	if longer than 0.25 sec & ave.duration in the phrase		2	
		else		1	
	most ascending & leaping note	if longer than 0.25 sec & ave.duration in the phrase	0	2	
		else	2	0	
misc	articulation	accent		1	
		tenuto		1	
		non-accidental		1	
		staccato		1	
	short slur (about 1 second or shorter)	beginning note		1	
		ending note		-1	
	rest	on strong beat & following note is on weak beat		0	1
		on weak beat & following note is on strong beat	1	0	

Table 1. Rules and parameters for calculating apex likelihood

ON THE TRACEABILITY OF THE COMPOSITIONAL PROCESS

Hanns Holger Rutz
University of Plymouth
Interdisciplinary Centre for
Computer Music Research (ICCMR)
hanns.rutz@plymouth.ac.uk

Eduardo Miranda
University of Plymouth
Interdisciplinary Centre for
Computer Music Research (ICCMR)
eduardo.miranda@plymouth.ac.uk

Gerhard Eckel
University of Music
and Performing Arts Graz
Institute of Electronic Music
and Acoustics (IEM)
eckel@iem.at

ABSTRACT

Composition is viewed as a process that has its own temporal dimension. This process can sometimes be highly non-linear, sometimes is carried out in real-time during a performance. A model is proposed that unifies the creational and the performance time and that traces the history of the creation of a piece. This model is based on a transformation that enhances data structures to become persistent. Confluent persistence allows navigation to any previous version of a piece, to create version branches at any point, and to combine different versions with each other. This concept is tuned to integrate two important aspects, retroactivity and multiplicities. Three representative problems are posed: How to define dependancies on entities that change over time, how to introduce changes ex-post that affect future versions, and how to continue working on parallel versions of a piece. Solutions based on our test implementation in the Scala language are presented. Our approach opens new possibilities in the area of music analysis and can conflate disparate notions of composition such as tape composition, interactive sound installation, and live improvisation. They can be represented by the same data structure and both offline and realtime manipulations happen within the same transactional model.

1. CONCEPTUAL FOUNDATION

1.1 The Double Nature of Composition

Time, along with space the fundamental parameter of composing pieces of music and sound art, appears in various forms. A fundamental distinction can be drawn between the *creational* time t_K – the time in which a composer creates or manipulates a piece – and *performance* time t_P – the time in which a piece is presented to a listener. One feels reminded of the double nature of the term *composition*, as pointed out by Koenig: «By musical composition we generally understand the production of an instrumental score or a tape of electronic music. However, we also understand composition as the result of composing [...] (we say for instance: "I have heard a composition by composer X").» [1, p. 191] The hermetic view that the «concept of composition is accordingly closed with regard to the result, but open with regard to the making of a composition» [1, ibid.], however is dropped in favour of one where the electroacoustic composer is regarded as the «first listener» [1] , often being able to "perform" the piece in total or part while working on it, or in fact performing it in a live improvisation where part of the work is composed (put together) while being performed, not necessarily arriving at one pre-defined result.

A special case is added by the medium of sound installation, where the "piece" often does not have a beginning or ending, and in which the listener chooses the time span of exposure to the sounds, sometimes even influencing the piece in an interactive way. It is therefore useful to depart from a perspective where the process of composition *terminates* in a fixed composition, but rather to consider indeterminacy as an essential ingredient, and therefore to look out for ways in which indeterminacy can be represented and manipulated in a composition system. Tentatively, we further divide elements in t_P into those forming a virtual (or prospective) structure and into those forming several actual realisations of that structure.

1.2 Databases

An interesting taxonomy has been developed in research on *database* systems: In a bi-temporal database, two timelines are distinguished: The *valid* time defines the time in which a database entry has existence in "reality", when it «accurately modeled reality» [3]. On the other hand, someone is maintaining and editing the database, performing operations on the transactional level, where time means «when an event is recorded in the database» [4]. The two timelines are often seen as orthogonal to each other, and common queries are punctiform with regard to transactional time and interval based with regard to valid time (cf. [5]). It follows that, if a composition is con-

[1] The term is deliberately taken from Daniel Charles who demands «a music that takes care to consider the composer not as the organizer of a technological ritual but, more modestly, as the first listener». [2, p. 152]

sidered being a kind of database, the composer takes the role of the operator on the database, and transactional time corresponds with t_{K}. The entries in the database, the elements from which the music is constructed, form one or more valid timeline fragments in virtual t_{P} and eventually become one particular actual timeline in t_{P} per performance.

1.3 Persistence

The extension of the valid by the transactional time layer introduces the history of the creation of a work. In the taxonomy of Driscoll et al., we call *ephemeral* a data structure which is agnostic of its history, so that any modification to it would let the previous state fall into oblivion. On the other hand, enhancing an ephemeral structure so that its previous states are still accessible, makes it become persistent [6]. The distinct variants of an ephemeral structure we call "versions", and the versions form vertices in a directed graph such that one vertex v_j points to another v_k if v_k was created by applying some modification to the ephemeral structure in version v_j.

In a linear perspective, each transaction corresponds with a new version of the database. However, in a non-linear perspective, one could depart from any previous version and branch off. Finally one could even create a version by combining two previous versions. In the first case, the graph is a linear path, and the enhanced structure is called *partially* persistent, meaning that «all versions can be accessed but only the newest version can be modified». The second case produces a graph which is a tree, and we speak of full persistence, where «every version can be both accessed and modified». In the last case the enhancement which uses «an ephemeral data structure that supports an update in which two different versions are combined» is called confluent persistence, and the versions have the most general form of a directed acyclic graph (DAG) [6].

We will use persistence not only to model the evolution of a piece in t_{K}, but also to use version branching and melding as a *joint* between virtual elements and one or more actual realisations in t_{P}. This way we unify the compositional process and the performance of pieces.

A critique of the persistence approach comes from Demaine et al. who state that, since versions are never overwritten and versions can only depend on previous versions, «the dependence relationship between two versions never changes. [...] Thus, the persistence paradigm is [...] inappropriate for when changes must be made directly to the past state of the structure.» [7] Instead they propose «retroactive data structures» as a new approach that can incorporate deliberate manipulations of past states of a data structure. Unlike persistence for which general transformations have been proposed, retroactivity requires special solutions for each particular data structure. In section 2.5 we will face a problem that seemingly calls for some kind of retroactivity, and we will see that it can well be solved within the persistence paradigm.

1.4 Multiplicities

If now the compositional process is seen as a sequence of decision-making or actualisation, we may integrate indeterminacy into this model. Indeterminacy can be attributed to three sources: Chance operations, interactive sensorial input, and generative (self-modifying or memorising) structures. Indeterminacy can be subsumed under other forms of multiplicities, namely the exploratory behaviour of the composer who concurrently or successively develops different versions of (parts of) a piece, and scale where different versions are developed for different contexts, e.g. modes of spatialisation. In total this leads to five types of multiplicities. The task is then to elaborate appropriate models for the virtual sources of multiplicities, for example a model of chance operations, a model of interactive input, etc. Although this is beyond the scope of this paper, we will use a simple placeholder for temporal values which are unknown in a version to show that our general model can indeed be extended to represent multiplicities.

2. IMPLEMENTATION

2.1 An Overview of Confluent Persistence

Fiat and Kaplan [8] have developed general algorithms for turning any ephemeral linked data structure into its confluently persistent counterpart. Our contribution is to apply this framework to the representation of temporal objects, extending it in several ways. Before we describe problem cases and their solution, it is therefore necessary to give a brief overview of this framework.

The ephemeral data structure is considered to be composed of any number of nodes each of which can have "data" and "pointer" fields. The pointer fields are used to link nodes together, while the data fields hold primitive values. Instead of distinguishing data and pointer fields, we prefer to speak about mutable fields used to store *immutable* values and mutable fields used to refer to other *mutable* objects.

Each ephemeral node is transformed into a "fat" node which contains information about all its states through different versions, using some kind of dictionary for each field. To distinguish which version we are looking at, the notion of a node pedigree is introduced which basically is a sequence of version identifiers (or vertices), starting from the version in which a node was created (the seminal version), and carrying successively the identifiers of the versions through which the node was brought to the currently accessed version. The identifier of a fat node thus is the tuple composed of the node pedigree and a reference to the fat node structure.

To access a field in a particular version of a fat node, each ephemeral field is replaced by a fat field

which consists of a *search trie* [2] that carries all values that have ever been assigned to that field, stored in the leaves of the trie, and the paths into the trie being the so-called assignment pedigrees, again a sequence of version identifiers. In the case of pointer fields – references to mutable objects –, node identifiers are stored in the trie, and in the retrieval a transformation called «Pedigree Prefix Substitution» is applied to update the node identifier so that it becomes a valid and unique access identifier within the current path in the version DAG.

The notion of pedigrees allows for the appearance of an ephemeral node more than once within the same version, while maintaining correct access to each element. The operation by which a node is re-introduced into a version is called a "meld", and it allows for example to catenate a linked list to itself or to an older version of itself.

Our implementation uses Fiat and Kaplan's *compressed-path* representation of pedigrees. It is based on the observation that there are often long linear sequences in the version graph which produce weak performance when using a full-path pedigree representation. In the full-path representation, the trie keys grow linearly with the number of versions. In the compressed-path method, the graph is split into disjoint (sub-)trees, each of which has an associated level ℓ, can only be entered at most once per path and will be represented by two symbols – the identifier of the version at which one enters the subtree and the identifier of the version from which one leaves the subtree (or the terminal version if the path ends in this subtree). A new subtree with an incremented level needs to be created when a meld operation is based on elements from versions of the same tree level. The so-called index $\tilde{c}(p)$ of the compressed path $c(p)$ is used now as key into the search trie of the fat fields. It contains all elements of $c(p)$ but the last, the particular version vertex inside one subtree. The value stored in the trie is a data structure that contains all the mappings from the target vertices of assignments (last elements of compressed paths) to the assigned values. Given a query key, this structure can find the nearest ancestor vertex in the subtree.

For the tries, we employ the lexicographic *splay* tree of Sleator and Tarjan [9], for the target vertex mapping we use a plain list along with a total ordering of the vertices imposed by the pre-order and post-order of the subtree, as suggested by Dietz [10] [3], although more sophisticated and better performing data structures are known (cf. [11]).

[2] A trie, also called prefix tree, is an associative data structure where the key consists of a sequence of elements. To find a value in the trie, the first element of the key is compared to the root node and the according branch is taken, then the second element is compared to the node at the second level, and so forth.

[3] Briefly, a vertex v_j is ancestor of another vertex v_k, if v_j appears left to v_k in the pre-order traversal and right to it in the post-order traversal of the tree. Among the candidates, the element that is rightmost in the pre-order list is the nearest ancestor.

2.2 Posing a Problem

We will introduce our approach by posing a simple problem, illustrated in Figure 1, that is to be solved.

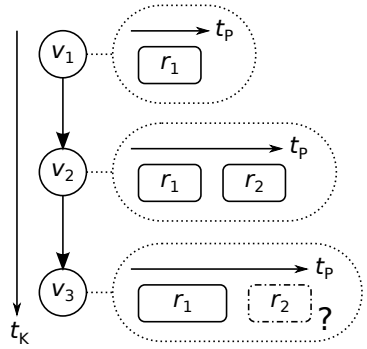


Figure 1. First problem: How to define the dependancy of region r_2 on r_1 such that the former automatically moves along with the latter in future versions?

Bi-temporality is modeled by associating each version vertex v_j with a point on the creational timeline (implicit), and the performance timeline t_P is expressed in the ephemeral data structure such that objects in time are associated with a time interval in t_P. Let the basic class of such objects be (fat) regions, denoted by r_j. A region can be anything from an audio file snippet to a code block that generates synthesised sound. Let $i(r_j)$ be the *interval* of a region, specified by a beginning time *start* and a duration *dur*, such that $i(r_j) = (start, dur)$. These points in time we call *periods*.

There are three operations performed by the composer. In v_1, a new region r_1 is created, forming the initial element of the performance timeline. The second operation, forming the next version v_2, adds a second region r_2 such that $i(r_2)$ starts at a fixed offset after $i(r_1)$ stops. Finally, the composer decides that r_1 should last longer and adjusts the duration of its interval accordingly. The problem is to define the relationship between r_1 and r_2 such that under the modification of r_1, we preserve the intention of r_2 following r_1.

We assume a set of arithmetic operations on intervals and periods, such as addition [4]. In v_2 we would say that $i(r_2)$ is created by an expression such as $(start = stop(i(r_1)) + p_o, dur = p_d)$ where p_o is some period offset and p_d some period duration. However, the manifest idea of applying the confluent persistence technique to either *resolve* the value of the fat interval field of r_1 at version v_2 or to create a fat pointer *reference* entry to it at version v_2 would create the fixed "dependence relationship" that was criticised by the retroactive approach – $i(r_2)$ would depend on $i(r_1)$ assigned in the version that is closest ancestor of v_2. What we want instead is to depend on this interval *no matter* at which version of $i(r_1)$ we are looking. The solution is to use a kind of dynamic reference.

[4] E.g., we simply define an interval's *stop* as $start + dur$.

Before we give this solution, we present our testing framework.

2.3 The Testing Framework

For our implementation, we use the Scala programming language [12]. Scala combines object-oriented and functional approaches and has a rich type system with single class inheritance and multiple mixins called *traits*. A trait can declare a set of abstract methods, but can also provide concrete definitions. Although Scala is a compiled language, we use a read-eval-print-loop (REPL) that allows one to create version vertices and navigate between them step-by-step. We also anticipate that in a composition environment based on this framework, the composer would typically create structures in a REPL. Furthermore, we provide access to the persistence sensitive environment in the form of an internal domain specific language extension which is realised by a combination of methods imported into the REPL scope and so-called implicit conversions, a language construct of Scala that can be used to seemingly enrich existing classes with new methods. For example, we add a method `secs` to the floating point class `Double` to create period literals. Figure 2 shows an overview of the classes involved.

The environment maintains two access paths into the version graph, one for reading and one for writing. When creating a new version, referencing and accessing existing objects involves the reading path which denotes the version we are departing from, and assignments are made using the writing path which denotes the newly created version. A single version step is performed by method `t[T](thunk: => T): T` which takes an argument *thunk* – a parameterless function with result type `T` – and evaluates it in a context where the read access corresponds to the current version, and the write access corresponds to a version newly derived from the current version. The first step in figure 1 becomes:

```
val r1 = t { region("r1", 0.secs :< 3.secs) }
```

Note how the interval literal is constructed by taking a *start* and a *dur* period and catenating them with the `:<` operator. The result of this operation is stored in value `r1`. It is a special region *handle* that the environment automatically converts to a region *identifier* when used. This saves us from explicitly updating each access identifier when navigating through the version graph.

2.4 Solution to the Problem

Assuming that method `interval` on the region object returns a reference with the desired semantics as requested in the conclusion of section 2.2, the code for versions v_2 and v_3 becomes:

```
val r2 = t { region("r2", (r1.interval.stop +
    2.secs) :< 5.secs) }
t { r1.interval = 0.secs :< 7.secs }
```

These semantics are achieved by constructing an `IntervalProxy` that wraps the underlying fat interval field in r_1. This proxy delegates the interval methods by using a special `access` method. When arithmetics are performed on the proxy's `start` or `dur` fields, they are wrapped in special `PeriodExpr` objects.

The `access` call performs a pedigree prefix substitution as in the pointer retrieval of [8], but *prior* accessing the fat field. As a consequence, the proxy acknowledges all modifications made to the field between the creation of the proxy and the current version. We therefore call this behaviour "fluent". In order to access a region's interval without the fluent behaviour, the proxy is fixed via `r1.interval.fixed` (cf. fig. 2). Note that linearity in t_K is maintained, so $i(r_2)$ in v_2 is still referring to $i(r_1)$ in version v_1!

2.5 Retroactive Operations

In the second problem, the approach of section 2.4 does not help, as we are now faced with version branching. The problem is depicted in figure 3.

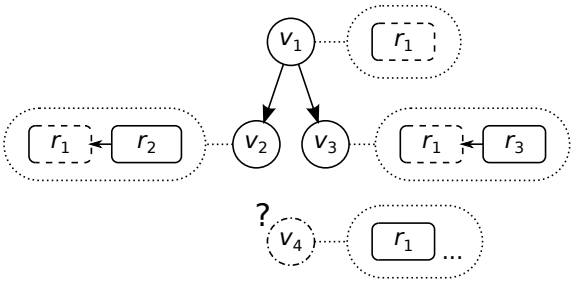


Figure 3. The second problem addresses changes that need to affect more than one future version.

Here the composer decided to begin with a region r_1 but leaves its duration subject to a later decision. Next, a version v_2 is produced by adding another region to start after region r_1 stops, similar to the previous example. The dependency of the regions is indicated by the arrow from r_2 to r_1. After completing version v_2, the composer tries out a different variant, starting over from version v_1, the result of which is version v_3. The finishing task, deciding on the final interval of r_1, seemingly calls for a retroactive operation – a correction of v_1. However, throughout our framework we pursue the preservation of causality, that is, the original states of $v_1 \ldots v_3$ must still be accessible. The solution is to insert a new vertex v_4 between v_1 and its former children, as shown in figure 4(a).

The data structure maintained for the nearest ancestor search within a subtree – and until now we have only dealt with graphs consisting of a single tree – can be tuned for such a quasi-retroactive insertion. Method `retroc` inserts a new version vertex (here v_4) right after its parent vertex (here v_1) in the pre-order traversal list, and right before it in the post-order traversal list. In this case, this yields a pre-order of $\langle v_1, \boldsymbol{v_4}, v_2, v_3 \rangle$ and a post-order of $\langle v_2, v_3, \boldsymbol{v_4}, v_1 \rangle$, satisfying the condition of v_4 being the closest ancestor

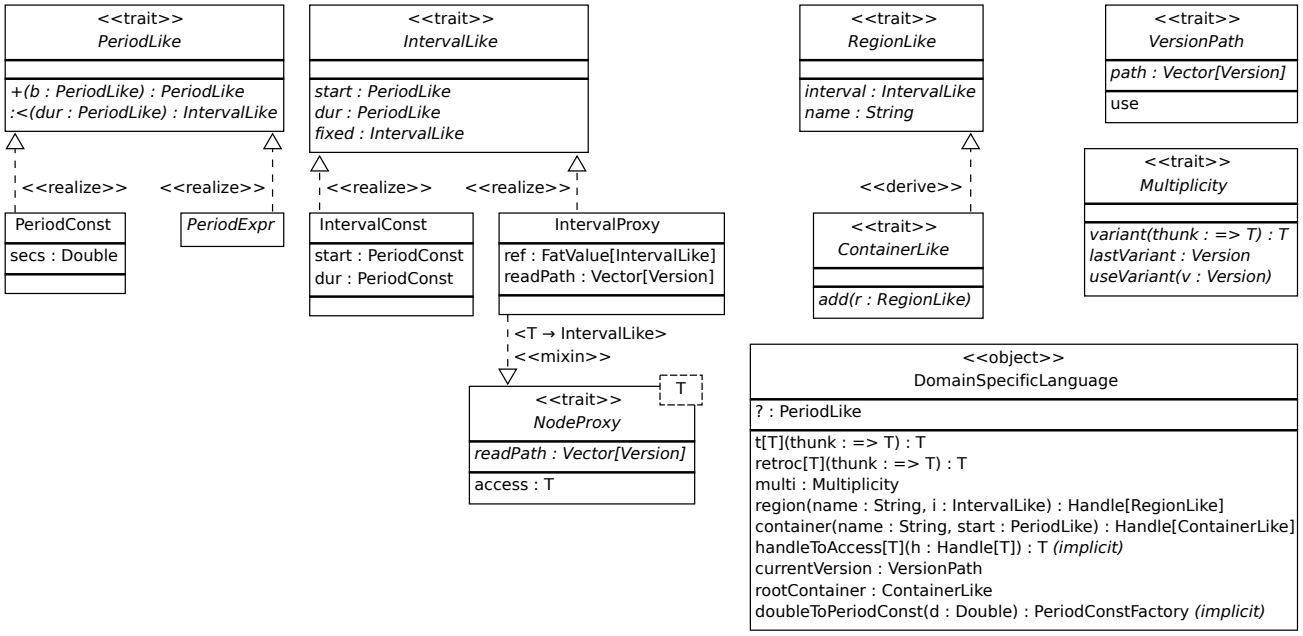


Figure 2. Class Diagram. Depicted are only those classes, attributes and methods referred to in this paper.

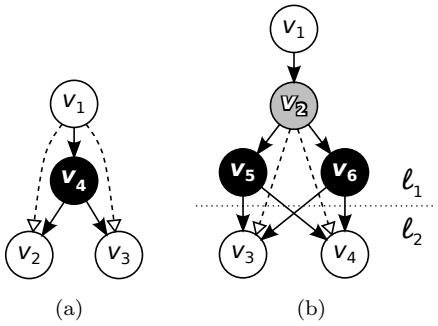


Figure 4. Retroactive vertices. (a) Single "corrective" operation. (b) Incorporating multiplicities.

of v_2 and of v_3 (cf. footnote 3). The corresponding code is:

```
val r1 = t { region("r1", 0.secs :< ?) }
val v1 = currentVersion
val r2 = t { region("r2", (r1.interval.stop +
    2.secs) :< 5.secs) }
v1.use
val r3 = t { region("r3", (r1.interval.stop +
    3.secs) :< 7.secs) }
v1.use // parent of the retroactive vertex
retroc { r1.interval = 0.secs :< 4.secs }
```

We use `?` as a placeholder for an unspecified period. The `currentVersion` method is used to capture the currently accessed version path. Navigation back to a particular version is achieved by calling `use` on a version path.

It has not been explained yet how the linearity of t_K is preserved, so that the original states of versions v_2 and v_3 are not lost. This is achieved by conditioning the ancestor lookup using the monotonically increasing vertex indices: When looking up a target vertex v_k, only vertices $v_j, j \leq k$ are considered. Therefore, v_4 becomes effective only after creating further descendants from the graph's leaves. For example, if v_5 is created from v_2, the modifications of v_4 will be effective in v_5 (since $4 \leq 5$), but not in v_2 (since $4 > 2$), preserving the original state of v_2. This conditional behaviour is indicated in figure 4(a): In v_2 and v_3 the dashed arrows are followed, while in descendants of v_2 and v_3 the solid arrows are effective.

Going back to the conceptual layout, intuitively one could think of using *repeated* retroactive insertions to represent the various outcomes of a multiplicity. For instance, if $v_5^\star$ was a new correction to $i(r_1)$, it would be inserted as a retroactive child vertex to v_4. However, the composer would not be able to freely *switch* between these two variants for $i(r_1)$ at a later point, precisely because we enforced the linearity in t_K.

We introduce another method `multi` which is dedicated to this problem. `multi` can only be executed on leaves of the graph, because it enforces a successive tree split. Figure 4(b) illustrates this: Initially, `multi` creates a *neutral* vertex v_2 which functions as a common ancestor for any future outcomes of the multiplicity. The versions v_3 and v_4 inserted after the multiplicity are enforced to start new subtrees at level ℓ_2. All vertices belonging to the multiplicity, located at the smaller tree level ℓ_1, will be explicitly included in the compressed path-representations as exiting vertices and can thus be seen as mutual switches. For instance, if we wish to access version v_3 incorporating variant v_5, the compressed path would be $\langle v_1, \boldsymbol{v_5}, v_3, v_3\rangle$, if variant v_6 was desired, the path would be $\langle v_1, \boldsymbol{v_6}, v_3, v_3\rangle$.

2.6 Melding and Parallel Motion of Versions

The enforcement of tree splitting in `multi` makes one think of the original meld operation. Indeed the vertices of the versions succeeding a multiplicity may have an indegree of > 1 (the indegree is the number of variants realised) which is also characteristic of a version meld. The difference is, that in the original confluent persistence framework, the set of access pointers to the data structure remains the same. With the use of multiplicities we are facilitating what we call "parallel motion" in the version graph so that there are different access paths to a particular vertex, carrying the information about which variant of each multiplicity is "active". This is clarified by a final example, shown in figure 5.

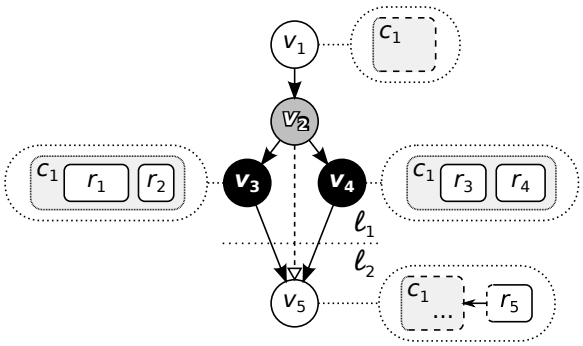


Figure 5. The third problem demands a facility to continue in a piece without settling on one particular version (v_3 versus v_4).

Here, the composer has planned a section for live improvisation and created a *container* c_1 for it in the first version. A multiplicity (neutral vertex v_2) is created to host the different improvisations, so one can later switch forth and back between them. The two variants v_3 and v_4 are simply modeled by adding regions to the container, as the framework really is agnostic to whether an operation is carried out offline or in realtime. As the composition finally carries on in v_5, this version automatically has a tree level incremented from ℓ_1 to ℓ_2. The following code simulates this:

```
val c1 = t { container("c1") }
val m = multi
c1.use
val (r1, r2) = m.variant {
    (region("r1", 0.secs :< 4.secs),
     region("r2", 5.secs :< 2.secs)) }
val v3 = m.lastVariant
val (r3, r4) = m.variant {
    (region("r3", 0.secs :< 2.5.secs),
     region("r4", 3.5.secs :< 3.secs)) }
val v4 = m.lastVariant
rootContainer.use
val r5 = t { region("r5", (c1.interval.stop +
    2.secs) :< 4.secs) }
m.useVariant(v3)
r5.interval.fixed // result: 9.secs :< 4.secs
m.useVariant(v4)
r5.interval.fixed // result: 8.5.secs :< 4.secs
```

As can be seen, method `variant { }` creates a new variant transaction, while `useVariant` updates the access path to include a particular variant. Since independent regions are created in the two variants, they cannot be used as references for v_5. This reference problem is solved by introducing the *container* object. Implicitly, in all the previous examples, the regions had been added to the default `rootContainer`. We explicitly revert to it for region r_5 by calling `rootContainer.use`.

The neutral vertex v_2 gains additional significance here: As r_5 is created, it is added to the root container. Assuming that containers are modeled using a list of the contained objects along with a field for the size of this list, assignments with compressed path $\langle v_1, \boldsymbol{v_2}, v_5, v_5\rangle$ are produced in the fat root container. If now variant v_3 is activated, the current access path becomes $\langle v_1, \boldsymbol{v_3}, v_5, v_5\rangle$. Consequently, a query for the number of objects in the root container would terminate in the trie at v_1, producing the wrong result, since c_1 was the only container in v_1.

This last problem is solved by enhancing the maximum prefix search in the trie such that if a vertex (here v_3) is not found and this vertex belongs to a multiplicity, it is replaced by the corresponding neutral vertex (here v_2) and the last splaying is repeated.

3. CONCLUSION

We have modeled and an implemented the music composition process as a confluently persistent data structure where the version DAG forms the creational timeline t_K, and the structure itself contains temporal objects relating to performance time t_P. We have enhanced this structure with two new operations `retroc` to incorporate quasi-retroactive decision making and `multi` to integrate realisation variants of the piece. The apriority of Allombert et al. [13] – "1. The compositional process: the composer builds his interactive score [...] 2. The performance process: the interactive score is no more edited" – becomes meaningless, as the realisations become part of "the piece", they re-build it.

The framework remains to be tested in a real-world and real-time application. For this, an efficient scheduler representation of the temporal objects is needed. Changes induced by retroactivity and switching between variants of a multiplicity need to be propagated (e.g., in some form of publisher-subscriber pattern), and cases where queries become invalid must be handled. This question of inconsistency has been investigated by Acar et al. [14].

4. REFERENCES

[1] G. M. Koenig, *Ästhetische Praxis*, vol. 3 of *Texte zur Musik*, ch. Kompositionsprozesse (1978),

pp. 191–210. Saarbrücken: PFAU Verlag, 1993.

[2] D. Charles, "Entr'acte: "Formal" or "Informal" Music?," *The Musical Quarterly*, vol. 51, pp. 144–165, Jan 1965.

[3] R. Snodgrass and I. Ahn, "A taxonomy of time databases," *ACM SIGMOD Record*, vol. 14, pp. 236–246, May 1985.

[4] G. Copeland and D. Maier, "Making smalltalk a database system," *ACM SIGMOD Record*, vol. 14, pp. 316–325, June 1984.

[5] B. Salzberg and V. J. Tsotras, "Comparison of access methods for time-evolving data," *ACM Computing Surveys*, vol. 31, pp. 158–221, June 1999.

[6] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan, "Making data structures persistent," *Journal of Computer and System Sciences*, vol. 38, pp. 86–124, Feb 1989.

[7] E. D. Demaine, J. Iacono, and S. Langerman, "Retroactive data structures," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 2, pp. 13:1–13:20, 2007.

[8] A. Fiat and H. Kaplan, "Making data structures confluently persistent," in *Proceedings of the 12th annual ACM-SIAM symposium on Discrete algorithms*, pp. 537–546, 2001.

[9] D. Sleator and R. E. Tarjan, "Self-adjusting binary search trees," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 652–686, 1985.

[10] P. F. Dietz, "Maintaining order in a linked list," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pp. 122–127, 1982.

[11] S. Alstrup, T. Husfeldt, and T. Rauhe, "Marked ancestor problems," in *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, p. 534, 1998.

[12] M. Odersky, P. Altherr, V. Cremet, I. Dragos, G. Dubochet, B. Emir, S. McDirmid, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, L. Spoon, and M. Zenger, *An Overview of the Scala Programming Language*, Technical Report LAMP-REPORT-2006-001. 2nd ed., 2006. Online: http://www.scala-lang.org/docu/files/ScalaOverview.pdf.

[13] A. Allombert, G. Assayag, M. Desainte-Catherine, and C. Rueda, "Concurrent constraints models for interactive scores," *Proceedings of Sound and Music Computing Conferences (SMC)*, pp. 14:1–14:8, 2006.

[14] U. A. Acar, G. E. Blelloch, and K. Tangwongsan, "Non-oblivious retroactive data structures," tech. rep., CMU-CS-07-169, Carnegie Mellon University, School of Computer Science, 2007.

TIDAL – PATTERN LANGUAGE FOR LIVE CODING OF MUSIC

Alex McLean and Geraint Wiggins
Centre for Cognition, Computation and Culture
Department of Computing
Goldsmiths, University of London

ABSTRACT

Computer language for the description of pattern has been employed for both analysis and composition of music. In this paper we investigate the latter, with particular interest in pattern language for use in live coding performance [1]. Towards this end we introduce Tidal, a pattern language designed for music improvisation, and embedded in the Haskell programming language.

Tidal represents polyphonic patterns as a time varying function, providing an extensible range of pattern generators and combinators for composing patterns out of hierarchies of sub-patterns. Open Sound Control (OSC) messages are used to trigger sound events, where each OSC parameter may be expressed as a pattern. Tidal is designed to allow patterns to be created and modified during a live coded performance, aided by terse, expressive syntax and integration with an emerging time synchronisation standard.

1. INTRODUCTION

When we view the composed sequence "abcabcabc…" we quickly infer the pattern "repeat abc". This is inference of hierarchy aiding memory of long sequences, prediction of future values and recognition of objects. Pattern pervades the arts; as Alfred Whitehead [2] eloquently puts it, "Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern." To our shame these words were background to Whitehead lambasting those taking quotes out of context, but nonetheless communicate a role of pattern supported here; one individual encodes a pattern and another decodes it, both actively engaged with the work while creating their own experience. In this paper we examine the encoding of pattern in particular, introducing Tidal, a computer language for encoding musical patterns during improvised live coding performances [1].

Pattern is everywhere, and the subject of musical pattern is a broad subject alone. The desire to capture musical patterns with machines goes back to well before electronic computers. For example, Leonardo da Vinci invented a hurdy gurdy with movable pegs to encode a pattern, and multiple adjustable reeds which transformed the pattern into a canon [3]. Hierarchies and heterarchies of repeating structure run throughout much of music theory, and computational approaches to music analysis, indexing and composition all have focus on discrete musical events and the rules to which they conform [4, §4.2]. From this we assert that the encoding of pattern is fundamental to music making. In the following we review support given to musical pattern making by computer language, and then introduce Tidal, a language for live improvisation of musical pattern. Before that we motivate the discussion through brief review of the practice of *live coding*, for which Tidal has been created.

2. LIVE CODING

Since 2003 an active group of practitioners and researchers [5] have been developing new (and rejuvinating old) approaches to improvising computer music and video animation; activity collectively known as *live coding* [6, 1, 7]. The archetypal live coding performance involves programmers writing code on stage, with their screens projected for an audience. The code is dynamically interpreted, taking on edits on-the fly without losing process state, so that no unwanted discontinuities in the output occur. Here a software development process is the performance, with the musical or visual work generated not by a finished program, but its journey of development from an empty text editor to complex algorithm, generating continuously changing musical or visual form along the way.

A key challenge set to live coders is to react quickly in musical response to other performers, or else on their own whim. This can be difficult due to a straight trade off in the level of abstraction they have chosen; while a traditional instrumentalist makes one movement to make one sound, a live coder makes many movements (key presses) in order to describe many sounds. It is in a live coder's interest to find highly expressive computer language that allows their ideas to be described succinctly. The subject of We believe the prese focus on the composition of pattern language provides a positive step in the right direction.

3. PATTERN LANGUAGE

Literature on pattern language is mainly concerned with *analysis* of composed works relative to a particular theory of music. For example Simon and Sumner [8] propose a formal language for music analysis, consisting of a minimal grammar for describing phrase structure within periodic patterns. Their language allows for multidimen-

sional patterns, where different aspects such as note value, onset and duration may be expressed together. The grammar is based on a language used for description of aptitude tests which treat pattern induction as a correlate for intelligence. Somewhat relatedly, research has since suggested that there is a causal link between music listening and intelligence [9], known as the "Mozart effect". However this result has proved highly controversial [10], and we would certainly not claim that pattern language makes you clever. Deutsch and Feroe [11] introduced a similar pattern language to that of Simon and Sumners, for the analysis of hierarchical relationships in tonal music with reference to gestalt theory of perception.

The analytical perspective shown in the above languages puts focus on simple patterns with unambiguous interpretation. Music composition however demands complex patterns with many possible interpretations, leading to divergent perception across listeners. Therefore pattern language for synthesis of music requires a different approach from analysis. Indeed, a need for the design of pattern language for music composition is identified by Laurie Spiegel in her 1981 paper "Manipulations of Musical Patterns" [12]. Twelve classes of pattern transformation, taken from Spiegel's own introspection as a composer are detailed: transposition (translation by value), reversal (value inversion or time reversal), rotation (cycle time phase), phase offset (relative rotation, e.g. a canon), rescaling (of time or value), interpolation (adding midpoints and ornamentation), extrapolation (continuation), fragmentation (breaking up of an established pattern), substitution (against expectation), combination (by value – mixing/-counterpoint/harmony), sequencing (by time – editing) and repetition. Spiegel felt these to be 'tried and true' basic operations, which should be included in computer music editors alongside insert, delete and search-and-replace. Further, Spiegel proposed that studying these transformations could aid our understanding of the temporal forms shared by music and experimental film, including human perception of them.

Pattern transformations are evident in Spiegel's own Music Mouse software, and can also be seen in music software based on the traditional studio recording paradigm such as Steinberg Cubase and Apple Logic Studio. However Spiegel is a strong advocate for the role of the musician programmer, and expresses hope that these pattern transformations would be formalised into programming libraries. Such libraries have indeed since emerged. Hierarchical Music Specification Language (HMSL) developed in the 1980s includes an extensible framework for algorithmic composition, with some inbuilt pattern transformations. The Scheme based *Common Music* environment, developed from 1989, contains a well developed object oriented pattern library [13]; classes are provided for pattern transformations such as permutation, rotation and random selection, and for pattern generation such as Markov models, state transition and rewrite rules. The SuperCollider language [14] also comes with a extensive pattern library, benefiting from an active free software development community, and with advanced support for live coding. These systems are all inspiration for our own pattern language, introduced below.

4. TIDAL

Tidal is a pattern language embedded in the Haskell programming language, consisting of pattern representation, a library of pattern generators and combinators, an event scheduler and programmer's live coding interface. This is an extensive re-write of earlier work introduced under the working title of *Petrol* [15]. Extensions include improved pattern representation and fully configurable integration with the Open Sound Control (OSC) protocol [16].

4.1 Features

Before examining Tidal in detail we first characterise it in terms of features expected of a pattern language.

4.1.1 Host language

Tidal is a domain specific language embedded in the Haskell programming language. The choice of Haskell allows us to use its powerful type system, but also forces us to work within strict constraints brought by its static types and pure functions. We can however turn this strictness to our advantage, through use of Haskell's pioneering type-level constructs such as functors and monads. Once the notion of a pattern is defined in terms of these constructs a whole range of cutting edge computer research becomes available, which can then be explored for application in describing musical pattern.

Tidal inherits Haskell's syntax which is both terse (thanks to its declarative approach) and flexible, for example it is trivial to define new infix operators. Terse syntax allows for faster expression of ideas, and therefore a tighter programmer feedback loop more suitable for creative tasks [17].

4.1.2 Pattern composition

In Tidal, patterns may be composed of numerous sub-patterns in a variety of ways and to arbitrary depth, to produce complex wholes from simple parts. This could include concatenating patterns time-wise, merging them so that they co-occur, or performing pairwise operations across patterns, for example combining two numerical patterns by multiplying their values together. Composition may be heterarchical, where sub-pattern transformations are applied at more than one level of depth within a larger pattern.

4.1.3 Random access

Both Common Music and SuperCollider represent patterns using lazy evaluated lists, where values are calculated one at a time as needed, rather than all together when the list is defined. This allows long, perhaps infinitely long lists to be represented efficiently in memory as generator functions, useful for representing fractal patterns for example. In some languages, including Haskell, lists are lazily evaluated by default, without need for special syntax. This is not how patterns are represented in Tidal however. Lazy lists

are practical for linear operations, but you cannot evaluate the 100th value without first evaluating the first 99. This is a particular problem for live coding; if you were to change the definition of a lazy list, in order to continue where you left off you must regenerate the entire performance up to the current time position. [1] Further, it is much more computationally expensive to perform operations over a whole pattern without random access, even in the case of straightforward reversal.

Tidal allows for random access by representing a pattern not as a list of events but as a function from time values to events. A full description is given in §4.2.

4.1.4 Time representation

Time can be conceptualised either as *linear change* with *forward order* of succession, or as a repeating cycle where the end is also the beginning of the next repetition [18]. We can relate the former to the latter by noting that the phase plane of a sine wave is a circle; a sine wave progresses over linear time, but its oscillation is a repeating cycle. As a temporal artform, the same division is present in music, in repeating rhythmic structures that nonetheless progress linearly. For this reason Tidal allows both periodic and infinite patterns to be represented.

Another important distinction is between discrete and continuous time. In music tradition, time may be notated within discrete symbols, such as Western staff notation or Indian bol syllables, but performed with subtle phrasing over continuous time. Tidal maintains this distinction, where patterns are events over discrete time steps, but may include patterns of floating point onset time deltas. More details on this in §5.1.

4.1.5 Ready-made generators and transforms

A pattern library should contain a range of basic pattern generators and transforms, which can be straightforwardly composed into complex structures. It may also contain more complex transforms, or else have a community repository where such patterns may be shared. Tidal contains a range of these, some of which are inspired by other pattern languages, and others that come for free from Haskell's standard library of functions, including its general support for manipulating collections.

4.1.6 Community

> "Computers're bringing about a situation that's like the invention of harmony. Sub-routines are like chords. No one would think of keeping a chord to himself. You'd give it to anyone who wanted it. You'd welcome alterations of it. Sub-routines are altered by a single punch. We're getting music made by man himself: not just one man." *John Cage, 1969* [19]

John Cage's vision has not universally met with reality, much music software is proprietary, and in the United States sound synthesis algorithms are impeded by software patents. However computer music languages are judged by their communities, sharing code and ideas freely, particularly around languages released as free software themselves. A pattern language then should make sharing abstract musical ideas straightforward, so short snippets of code may be easily used, modified and passed on. This is certainly possible with Tidal, although this is a young language which has not yet had a community grow around it. Towards this end however, the first author is developing a website for sharing snippets of musical code, for Tidal and other languages.

4.2 Representation

We now turn to the detail of how Tidal represents patterns. The period of a pattern – the duration at which it repeats – is represented in Haskell's type system as an integer:

```
type Period = Maybe Int
```

The integer type `Int` is encapulated within the `Maybe` type, so that we can represent both periodic and non-periodic (i.e. infinite) patterns. For example the pattern "*a* followed by repeating *b*s" has a `Period` of `Nothing`, and "*abcdefgh*, repeated" would have a `Period` of `Just 8` [2].

The structure of a pattern is defined as a function from integer time to a list of events:

```
type Behaviour a = Int → [Maybe a]
```

The name of the `Behaviour` type is borrowed from reactive programming nomenclature [20], where a behaviour is the term for a time-varying value. Note that `Behaviour` is an abstract type, where `a` is a wild card standing for any other type. For example a pattern of musical notes could be of type `Behaviour String`, where pitch labels are represented as character strings, or alternatively of type `Behaviour Int` for a pattern of MIDI numbered note events. Another thing to note is that the `Maybe` type is again employed so that non-values may be included in a list of events. The reader may ask, why would you want to store non-values in a list at all? We might simply answer that a rest has a particular musical identity and so needs to be represented. More practical motivation is shown in §5, where `Nothing` is shown to have different meaning in different situations.

A pattern then is composed of a `Behaviour` and `Period`, given the field names `at` and `period` respectively:

```
data Pattern a =
  Pattern {at :: Behaviour, period :: Period}
```

A pattern may be constructed as in the following example representing the repeating sequence "0, 2, 4, 6":

```
p = Pattern {at = λn → [Just ((n `mod` 4) * 2)],
             period = Just 4}
```

We access values by evaluating a behaviour with a time value, for example with the above pattern, `at p 1` evaluates to `[Just 2]`. As this is a cyclic pattern of period 4, `at p 5` would give the same result, as would `at p (-3)`.

[1] SuperCollider supports live coding patterns using PatternProxiess [7]. These act as place-holders within a pattern, allowing a programmer to define sub-patterns which may be modified later.

[2] `Just` and `Nothing` are the two constructors of Haskell's `Maybe` type

The above pattern is expressed as a function over time. An approach more idiomatic to Haskell would be to define it recursively, in this case defining `at p 0` to return `Just [0]` and subsequent `at p n` to return the value at `n - 1` plus two. However great care must be taken when introducing such dependencies between time steps; it is easy to produce uncomputable patterns, or as in this case patterns which may require whole cycles to be computed to find values at a single time point.

4.3 Pattern generators

A pattern would not normally be described by directly invoking the constructor in the rather long-winded manner shown in the previous section, but by calling one of the pattern generating functions provided by Tidal. These consist of generators of basic repeating forms analogous to sine, square and triangle waves, and a parser of complex sequences. The `sine1` function produces a sine cycle of floating point numbers in the range 0 to 1 with a given period, here rendered as grey values with the `drawGray` function:

```
drawGray $ sine1 16
```

Tidal is designed for use in live music improvisation, but is also applicable for off-line composition, or for non musical domains. We take this opportunity to illustrate the examples in the following sections with visual patterns of colour as above, in sympathy with the present medium. For space efficiency the above cyclic pattern is rendered as a row of blocks, but ideally would be rendered as a circle, as the end of one cycle is also the beginning of the next.

Linear interpolation between values, somewhat related to musical glissandi, is provided by the `tween` function:

```
drawGray $ tween 0.0 1.0 16
```

If a pattern is given as a string, it is parsed according to the context, made possible through Haskell's type inference, and a string overloading extension.

```
draw "black blue lightgrey"
```

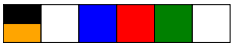
In the above example the `draw` function requires a colour pattern, and so a parser of colour names is automatically employed. Tidal can parse the basic types `String`, `Bool`, `Int` and `Float` and it is straightforward to add more as needed. All these parsers are expressed in terms of a common parser, which provides syntax for combining sub-patterns together into polymetric patterns. Sub-patterns with different periods may be combined either by repetition or by padding. In both cases the result is a combined pattern with period of the lowest common multiple of those of the constituent patterns. Combining patterns by repetition is denoted by square brackets, where constituent parts are separated by commas. In the following example the first part is repeated twice and the second thrice:

```
draw "[black blue green, orange red]"
```

Note that co-occurring events are visualised by the draw function as vertically stacked colour blocks.

Combining by padding each part with rests is denoted with curly brackets, and inspired by the Bol Processor [21]. In this example the first part is padded with one rest every step, and the second with two rests:

```
draw "{black blue green, orange red}"
```



In the above example there are steps where two events co-occur, and the block is split in two, where one event occurs, taking up the whole block, and where no events occur and the block is blank.

Polymetries may be embedded to any depth (note the use of a tilde to denote a rest):

```
draw "[{black ~ grey, orange}, red green]"
```

4.4 Pattern combinators

If an underlying pattern representation were to be a list, a pattern transformer would have to operate directly on sequences of events. For example, we might *rotate* a pattern one step forward by *pop*ping from the end of the list, and *unshift*ing/*cons*ing the result to the head of the list. In Tidal, because a pattern is a function from time to events, a transformer may manipulate time as well as events. Accordingly the Tidal function `rotL` for rotating a pattern to the left is straightforwardly defined as:

```
rotL p n =
  Pattern (λt → at p (t + n)) (period p)
```

Rotating to the right is simply defined as the inverse:

```
rotR p n =  rotL p (0 - n)
```

We won't go into the implementation details of all the pattern transformers here, suffice to say that they are all implemented as composable behaviours. The reader may refer to the source code for further details.

The `cat` function concatenates patterns together timewise:

```
drawGray $ cat [tween 0 1 8, tween 1 0 8]
```

As you might expect, the period of the resulting pattern will be the sum of the constituent pattern periods, unless one of the constituents is infinite, in which case the result will also be infinite.

A periodic pattern may be reversed with `rev`:

```
drawGray $ rev (sine1 8)
```

Or alternatively expressed forwards and then in reverse with `palindrome`:

```
drawGray $ palindrome (sine1 8)
```

The `every` function allows transformations to be applied to periodic patterns every n cycles. For example, to rotate a pattern by a single step every third repetition:

```
draw $ every 3 (1 `rotR`) "black grey red"
```

The Pattern type is defined as an applicative functor, allowing a function to be applied to every element of a pattern using the `<$>` functor map operator. For example, we may add some blue to a whole pattern by mapping the `blend` function (from the Haskell Colour library) over its elements:

```
draw $ blend 0.5 blue <$> p
  where p = every 3 (1 `rotR`) "black grey red"
```

If we were doing something similar to a sound rather than colour event, we might understand it as a musical transposition. We can also apply the functor map conditionally, for example to transpose every third cycle:

```
drawGray $ every 3 ((+ 0.6) <$>) "0.2 0.3 0 0.4"
```

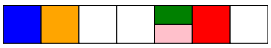
The Haskell applicative functor syntax also allows a new pattern to be composed by applying a function to combinations of values from other patterns. For example, the following gives a polyrhythmic lightening and darkening effect, by blending values from two patterns:

```
draw $
  (blend 0.5) <$> "red blue" <*>
"white white black"
```

The use of `<*>` here deserves some explanation. It allows us to map a function over more than one pattern at a time. In the above example, for each call to blend, a value is taken from each pattern. The `<*>` operator is defined for Patterns so that all events are used at least once, and no more than necessary to fulfil this constraint. Operationally, the shorter list of events is repeated until it is the same length as the longer; this is behaviour halfway between that of a Haskell `List` and a `ZipList`. For implementation details please refer to the code, but the end result are minimal combinations of polyphonic events without discarding any values.

The Tidal `onsets` function filters out elements that do not begin a phrase. Here we manipulate the onsets of a pattern (blending them with red), before combining them back with the original pattern.

```
draw $ combine [blend 0.5 red <$> onsets p, p]
  where p = "blue orange ~ ~ [green, pink] red ~"
```



The `onsets` function is particularly useful in cross-domain patterning, for example taking a pattern of notes and accentuating phrase onsets by making a time onset and/or velocity pattern from it.

5. OPEN SOUND CONTROL PATTERNS

Tidal has no capability for sound synthesis itself, but instead represents and schedules patterns of OSC messages to be sent to a synthesiser. Below we see how the 'shape' of an OSC message is described in Tidal:

```
synth = OscShape {path = "/trigger",
                  params =
                    [ F "note" Nothing,
                      F "velocity" (Just 1),
                      S "wave" (Just "triangle")
                    ],
                  timestamp = True
                 }
```

This is a trivial `"/trigger"` message consisting of two floating point parameters and one string parameter. Each parameter may be given a default value in the `OscShape`; in this case velocity has a default of `1`, wave has a default of `"triangle"` and note has no default. This means if a OSC pattern contains a message without a note value set, there will be no value to default it to, and so the message is discarded. Pattern accessors for each parameter are defined using names given in the `OscShape`:

```
note     = makeF synth "note"
velocity = makeF synth "velocity"
wave     = makeS synth "wave"
```

5.1 Scheduling

As `timestamp` is set to `True` in our OscShape example, one extra pattern accessor is available to us, for onset deltas:

```
onset = makeT synth
```

This allows us to make time patterns, applying subtle (or if you prefer, unsubtle) expression. This is implemented by wrapping each message in a timestamped OSC bundle. A simple example is to vary onset times by up to 0.02 seconds using a sine function:

```
onset $ (* 0.02) <$> sine 16
```

Instances of Tidal can synchronise with each other (and indeed other systems) via the NetClock protocol (`http://netclock.slab.org/`). NetClock is based upon time synchronisation in SuperCollider [14]. This means that time patterns can notionally schedule events to occur

in the past, up to the SuperCollider control latency, which has a default of 0.2 seconds.

It is also possible to create tempo patterns to globally affect all NetClock clients, for example to double the tempo over 32 time steps:

```
tempo $ tween 120 240 32
```

5.2 Sending messages

We connect our OSC pattern to a synthesiser using a `stream`, passing the network address and port of the synthesiser, along with the `OscShape` we defined earlier:

```
s ← stream "127.0.0.1" 7770 synth
```

This starts a scheduling thread for sending the messages, and returns a function for replacing the current pattern in shared memory. Patterns are composed into an OSC message Pattern and streamed to the synthesiser as follows:

```
s $ note ("50 ~ 62 60 ~ ~")
    ~~ velocity foo
    ~~ wave "square"
    ~~ onset ((* 0.01) <$> foo)
  where foo = sine1 16
```

The `~~` operator merges the three parameter patterns and the onset pattern together, into a single OSC message pattern. This is then passed to the stream `s`, replacing the currently scheduled pattern. Note that both `velocity` and `onset` are defined in terms of the separately defined pattern `foo`.

5.3 Use in music improvisation

Music improvisation is made possible in Tidal using the dynamic Glasgow Haskell Compiler Interpreter (`http://www.haskell.org/ghc/`). This allows the musician to develop a pattern over successive calls, perhaps modifying the preceding listing to transpose the note values every third period, make a polyrhythmic pattern of wave shapes, or combine multiple onset patterns into a chorus effect. Tidal provides a mode for the iconic emacs programmer's editor (`http://www.gnu.org/software/emacs/`) as a GHCI interface, allowing patterns to be live coded within an advanced developers environment. [3]

6. CONCLUSION

We have introduced Tidal, a language designed for live coding of musical pattern. Tidal has already been field tested through several performances by the first author, including to large audiences at international music festivals, informing ongoing development of the system. The system will be tested further through a series of planned workshops with potential users, and full documented release of the code, which is already available in its present form at `http://yaxu.org/tidal/`. A research programme is planned towards the development of a Graphical User Interface for live musical pattern making, with Tidal providing the pattern language. Work is ongoing towards applying Tidal to the domain of live video animation, with current focus on colour transitions over time inspired by the inventions of Mary Hallock-Greenwalt [22]. As mentioned in §4.1.6, we hope that a website for sharing ideas and code for musical patterning will encourage connections between communities of musicians and programmers interested in pattern.

[3] Projecting the emacs interface as part of a live coding performance has its own aesthetic, having a particularly strong effect on many developers in the audience, either of elation or revulsion.

7. REFERENCES

[1] N. Collins, A. McLean, J. Rohrhuber, and A. Ward, "Live coding in laptop performance," *Organised Sound*, vol. 8, no. 03, pp. 321–330, 2003.

[2] A. N. Whitehead, *Dialogues of Alfred North Whitehead (A Nonpareil Book)*. David R Godine, August 2001.

[3] L. Spiegel, "A short history of intelligent instruments," *Computer Music Journal*, vol. 11, no. 3, 1987.

[4] R. Rowe, *Machine Musicianship*. The MIT Press, March 2001.

[5] A. Ward, J. Rohrhuber, F. Olofsson, A. McLean, D. Griffiths, N. Collins, and A. Alexander, "Live algorithm programming and a temporary organisation for its promotion," in *read_me — Software Art and Cultures* (O. Goriunova and A. Shulgin, eds.), 2004.

[6] A. Blackwell and N. Collins, "The programming language as a musical instrument," in *Proceedings of PPIG05*, University of Sussex, 2005.

[7] J. Rohrhuber, A. de Campo, and R. Wieser, "Algorithms today: Notes on language design for just in time programming," in *Proceedings of the 2005 International Computer Music Conference*, 2005.

[8] H. A. Simon and R. K. Sumner, "Pattern in music," pp. 83–110, 1992.

[9] F. H. Rauscher, G. L. Shaw, and C. N. Ky, "Music and spatial task performance," *Nature*, vol. 365, p. 611, October 1993.

[10] K. M. Steele, K. E. Bass, and M. D. Crook, "The mystery of the mozart effect: Failure to replicate," *Psychological Science*, pp. 366–369, July 1999.

[11] D. Deutsch and J. Feroe, "The internal representation of pitch sequences in tonal music.," *Psychological Review*, vol. 88, pp. 503–22, November 1981.

[12] L. Spiegel, "Manipulations of musical patterns," in *Proceedings of the Symposium on Small Computers and the Arts*, pp. 19–22, 1981.

[13] H. K. Taube, *Notes from the Metalevel: Introduction to Algorithmic Music Composition*. Lisse, The Netherlands: Swets & Zeitlinger, 2004.

[14] J. McCartney, "Rethinking the computer music language: Supercollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[15] A. McLean and G. Wiggins, "Petrol: Reactive pattern language for improvised music," in *Proceedings of the International Computer Music Conference*, June 2010.

[16] A. Freed and A. Schmeder, "Features and future of open sound control version 1.1 for nime," in *NIME*, 2009.

[17] A. McLean and G. Wiggins, "Bricolage programming in the creative arts," in *22nd Psychology of Programming Interest Group*, 2010.

[18] G. Buzsaki, *Rhythms of the Brain*. Oxford University Press, USA, 1 ed., August 2006.

[19] J. Cage, *Art and Technology*. Cooper Square Press, 1969.

[20] C. Elliott, "Push-pull functional reactive programming," in *Haskell Symposium*, 2009.

[21] B. Bel, "Rationalizing musical time: syntactic and symbolic-numeric approaches," in *The Ratio Book* (C. Barlow, ed.), pp. 86–101, Feedback Studio, 2001.

[22] M. H. Greenewalt, *Nourathar, the Fine Art of Light Color Playing*. Philadelphia. Pa. Westbrook, 1946.

IN A CONCRETE SPACE. RECONSTRUCTING THE SPATIALIZATION OF IANNIS XENAKIS' *CONCRET PH* ON A MULTICHANNEL SETUP

Andrea Valle
CIRMA - Università di Torino
andrea.valle@unito.it

Kees Tazelaar
Institute of Sonology - Royal Conservatory
The Netherlands
info@keestazelaar.com

Vincenzo Lombardo
CIRMA - Università di Torino
vincenzo@di.unito.it

ABSTRACT

Even if lasting less than three minutes, Iannis Xenakis' *Concret PH* is one of the most influential works in the electroacoustic domain. It was originally created to be diffused in the Philips Pavilion, designed by the same Xenakis for the 1958 World Fair in Brussels. As the Pavilion was dismantled in 1959, the original spatialization design devised from the Pavilion has been lost. The paper presents new findings about the spatialization of *Concret PH*. It discusses them in the light of Xenakis' aesthetics, and consequently proposes a plausible reconstruction of the spatialization design. Finally, it proposes a real-time, interactive implementation of the reconstructed spatialization, rendered on a 8-channel setup using a VBAP technique.

1. INTRODUCTION

In 1956 Iannis Xenakis was working in the studio of Le Corbusier, when Philips company commissioned the famous architect a pavilion for the 1958 World Fair in Brussels [1]. The fair, being the first after the II World War, was a crucial event for the company: in particular, Louis Kalff, artistic director of Philips, considered it an occasion not to be renounced in order to show the world the technological advancements of the Dutch company. Le Corbusier accepted the commission and replied by promising to realize not an exhibit structure but a revolutionary "electronic poem". Le Corbusier's conception strictly adhered to the modernistic assumption that sees in technology the way in which art can fulfill a palingenesis of humanity: the architect proposed Philips a Wagnerian total artwork of sound and lights, taking place in a space explicitly designed as a container for the show. As a consequence, the project for the Philips Pavilion resulted in a complex work of art, the *Poème électronique*: an 8-minute multimedia work in which architecture, image and sound were deeply intermingled. The show included a black and white film, made of two filmed sequences created from still images, various light effects over the whole space, and electronic music to be delivered onto a multichannel system. While keeping for himself the creation of the visual part of the show, Le Corbusier asked one of the most avantgardist composers of XX-th Century to join the project, Edgar Varèse. In the occasion, Varèse created, as a musical counterpart for the visual component, his *Poème électronique*: originally a 3-track tape music, Varèse's *Poème* is one of the undisputed masterpiece of electronic music. At that time, Iannis Xenakis was an associate at Le Corbusier's studio, where he had already developed some of his well-known architectural exploits (e.g. the monastery of La Tourette). Xenakis was responsible for the design of the space. Xenakis turned Le Corbusier's original idea of a shell-like structure, based on a stomach-shaped plant, into a self-carrying, concrete shell, higher than 20 meters. More, the Pavilion's shape was generated by Xenakis as rule-based surfaces, namely hyperbolic paraboloids: the resulting shape was a tridimensional architectural object made of continuous curved lines. By explicit admission of Xenakis, the ruled surfaces of the Pavilion (see Figure 1) bear a structural relation to the striking opus 1 of the composer/architect, *Metastaseis* (1953/54) [2]. In this work, Xenakis started from a theoretical problem, that of defining a continuous transition between two discrete states (Xenakis, cited in [3], also [4], p. 32). The solution was based on devising a system of string glissandos with different speeds and ranges ("sonic spaces of continuous evolution", [5], p. 10). While designing the pavilion, his "inspiration was pin-pointed by the experiment with *Metastaseis*", so that there is a "causal chain of ideas" connecting the two works. Thus, in the Philips

Figure 1. The Philips Pavilion.

[1] This work extends the EU-funded VEP Project (http://edu.vrmmp.it/vep/), that has reconstructed the Philips Pavilion and the *Poème électronique* using virtual reality techniques. For a presentation of the project, including previous works, see [1].

Pavilion, "music and architecture found an intimate connection" ([5], p. 10). The internal surfaces of the Pavilion, covered with asbestos, were then literally encrusted with loudspeakers (for a total of 350). Loudspeakers were organized into "sound routes" (allowing sound to travel the space) and "clusters" (groups of contiguous loudspeakers playing together). Their presence converted the Pavilion into a "sounding room" ([6], 210), where, after more than 30 years, Varèse was able to finally listen to his music "literally projected into space" (Varèse, cited in [7]). Apart from the link with *Metastaseis*, the Philips Pavilion includes a second relevant element in relation to Xenakis' music. As a composer, Xenakis was allowed by Le Corbusier to create a short piece that should act as an interlude between two performances of the show: the 8 minutes of the *Poème électronique* were embedded in a cyclic program of 10 minutes, which was supposed to run continuously. The two additional minutes were reserved for an intermission, which would enable one audience to leave the pavilion while the next to enter, the "Interlude sonore". The *Interlude* would have then entered Xenakis' catalogue with the name of 'Concret PH". Because of its quite singular sonic structure (see later), the piece has gained a consistent fame, far beyond the specialists of contemporary and electronic music, and has been hailed as a precursor of "electronica" ([8], see also [9]). While discussing the piece, Harley observes that "the mobile sound trajectories throughout the Philips Pavilion would have no doubt been astonishing" ([10], p. 19). Still its original relation with the space of the Pavilion remains unclear. In the next sections, we first take into account newly available information on the *Interlude/Concret PH* from unpublished sources; then we discuss issues related to its spatialization in the Philips Pavilion and propose a novel reconstruction from unpublished sources; finally, we describe a simulation of the spatialization implemented on a 8-channel setup, presented publicly on January, 15th, 2010 at the EMF Foundation in New York, in the occasion of the the exhibition "Iannis Xenakis. Composer, Architect, Visionary" at The Drawing Center [11].

2. BEFORE *CONCRET PH*: THE *INTERLUDE*

In this section we reconstruct the history of the *Interlude* merging information from already published sources with new data coming from unpublished letters by Louis Kalff (now at the Philips Archive and in the private collection of Peter Wever, co-author of a historical study of the Brussels Expo [12]). The *Interlude* music was composed by Iannis Xenakis, and resembled his later published work *Concret PH*, but was not identical to it. The title "Concret PH" did not appear once in the correspondence relating to the design of the pavilion or in the official credits. On the plate near the entrance of the pavilion it was called "Interlude Sonore", which was also the title under which it was mentioned in the book *Poème électronique* released by Le Corbusier's collaborator Jean Petit [13]. An original version of the *Interlude Sonore* was found back in the archive of the Institute of Sonology in The Netherlands. It consists of three tracks and has a duration of 1'52". From the instructions Le Corbusier gave to Xenakis for the intermission music on 27 November 1957, one indeed gets the impression that he had little knowledge of the importance of Xenakis as a composer: "I have thought very seriously about your two minutes of music. What is it about? [...] it is a sort of carnival hawking, in which it is possible to pack a lot of wit and content that can touch a crowd that by definition is inattentive." ([6], p. 205). The letter ended with a very firm refusal to Xenakis' request to leave the office for three weeks to work on his piece in Eindhoven, using the same advanced equipment as Varèse was using for the *Poème*. Two days later, Le Corbusier sent a diagram of the intermission to Kalff which gave a description of Xenakis' music, very close to the music as we know it now: "Clouds of intermittent sounds, varying in density and intensity, and moving within the space of the pavilion."([6], p. 206). On 2 December 1957, Xenakis wrote to Kalff, describing the character of the intermission music as "sober, surprising and of an artistic quality, and at least as good as the rest of the spectacle". As in the case of *Poème électronique*, *Concret PH* was to be delivered inside the Pavilion, where loudspeakers were arranged into sound routes and cluster. Figure 2 shows a diagram of loudspeaker organization, from an original sketch by Xenakis. Sound routes are indicated in the text, clusters are marked with letters A, B, C, D, E, U and J. In order to spatialize sounds, a complex, totally automatic, routing system was devised, based on selectors as used in automatic telephone exchanges, switching on and off the loudspeakers [14] [1]. In the case of *Poème électronique*, a spatialization score was written. It includes instructions about *when* to drive a certain *track* into a certain *loudspeaker group*. If the group is a cluster, all the composing loudspeakers are activated/disactivated at the same time. If it is a sound route, then a stepping rate is indicated (i.e., the rate at which the sound travels along a route), in stepping impulses per second. In the sound routes, a sound "moves" as it is progressively routed from a speaker to the following one. The score was then translated into signals for controlling the dialers. Signals were recorded on a control tape to be played during the performance, driving the audio system. In a letter to Kalff from 11 March 1958, Xenakis gave a clear description of the material of the *Interlude Sonore* and the way it was supposed to be spatialized in the pavilion was given: "We recorded the sound of charcoal, and it is very beautiful. Please answer the following questions as soon as possible: a) will I be able to utilise three independent (but synchronised) tracks? b) will I be able to determine simultaneously the speed of movement along the horizontal belt?". Xenakis then explicitly indicates that each tracks was characterized in his intentions by a specific "sound" and a specific "speed" [15]. He also included a drawing (Figure 3), where sounds were indicated with Roman letters a, b, c and speeds with Greek letters α, β, γ. Kalff answered two days later: "We can [...] use only one track to reproduce your composition. The sound of the tape will be spread along the horizontal circle of loudspeakers surrounding the pavilion at a height of 3 to 4 metres. Along this circle, which will have approximately

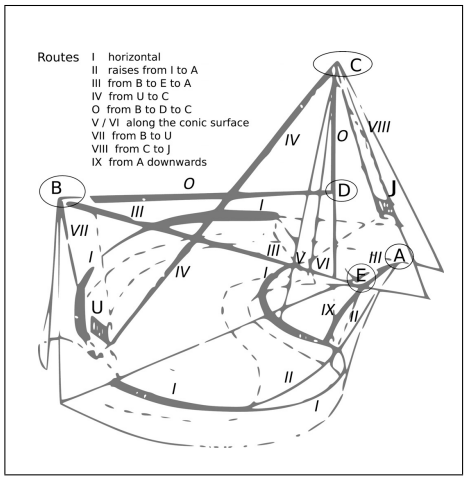


Figure 2. Diagram of clusters and routes of loudspeakers, adapted from Xenakis' original sketch (from [1]).

100 loudspeakers [2], the sound could move and stop at the points of your choice. [...] The two other synchronised tapes could then only be used to reproduce other simultaneous sounds from other directions" [15]. It also becomes clear from the technical description of the whole system in Philips Technical Review [14] that it would be impossible to feed three different sound sources to one 'route du son, let alone having them travel at different speeds in different directions. More, a route could only stand or move in one direction. In Varèse's score for *Poème électronique*, routes never stands, but always move: otherwise, indeed, they would simply act as clusters. Xenakis, obviously disappointed, wrote back on 17 March 1958: "With this letter I include a copy of the two minutes score. This score was conceived for three tapes and for high speeds of movement. You now tell me that I have only one track at my disposal [for the movement]. I am sorry to hear that, but I will adapt myself. But then I would like to know if the conceived speeds in the score are possible, and on the other side if the technicians will have sufficient time to make all the necessary connections. May I ask you for your opinion on the copy I send you? This would be the best way for me to find out about the technical limitations. Please send me this as soon as possible" (17 March 1958) [15]. But then in the end, Tak, the technical supervisor for sound-related aspects in the Pavilion, seemed to have come up with a solution: "We believe that it would be more fascinating if your sounds would be reproduced with all available means. This is why we asked our technicians to develop the equipment in such a way that the machines with three tracks and the machines for the automation can be used during the intermission as well. I have the pleasure to tell you that they managed to do that and that you have three individual tracks at your disposal. What you demand in the excerpt of your composition will be possible to realise." (24 March 1958) [15]. Xenakis then sent the three tapes and showed his gratitude for being allowed to use the spatial possibilities of the sound system in the pavilion: "By the end of this week I will send the tapes and the score of the two minutes of the Interlude, and my answer to the nice letter of Tak, who together with you, authorises me to use the third dimension." (April 3rd, 1958) [15]. Four days later, in another letter to Kalff, Xenakis continued: "I think that in the meantime you should have well received the three tapes of the music for the Interlude. I hereby include the stereophonic [sic] score and a letter to Tak. [...] The three tapes will give you an impression of the sounds and their composition (articulation). New effects will be created in three dimensions. Despite their identity, the uniformity of the used sounds will not disturb the timbral richness employed by Varèse. I thus stayed in my very secondary role. I have only used the third dimension according to your instructions. If that would be too disturbing, we could return to the horizontal zone. I have added another 30 seconds, because I don't think two minutes will be enough to empty and fill the building, even if we would use tear gas (and by the way, why not?)" [15]. Kalff replied to Xenakis on 9 April 1958: "We have received your tapes in good order. We have given them to Tak, together with the score. He is working day and night to complete the composition of Varèse with the three-dimensional effects" [15]. When Xenakis was able to hear his piece inside of the pavilion for the first time is not completely clear, but in a letter on 18 April 1958 to Varèse, he showed his dissatisfaction with the quality of the tapes [15]. According to a report by Kalff, efforts to improve the playback quality of the *Interlude Sonore* were undertaken between 24 April and 2 May 1958: "Mr. Tak will go to Eindhoven to process the sound tapes of Xenakis for the intermission [...] He will be finished by the end of the week and will bring them to Brussels either the 28th or the 29th" [3]. After having heard the complete program in the Pavilion for the first time on 2 May 1958, Xenakis was still not happy because the tapes had been remastered in Eindhoven at a level below his specifications and his likings. Kalff acknowledged the problem, and in a letter from 28 July 1958, Xenakis was invited to go to Eindhoven and work on the revisions in August; the final versions, however, would have been completed only in early August [6]. Indeed, the letters show a sort of continuous reworking process of the piece, both concerning sound material and spazialization.

3. SPATIALIZATION RECONSTRUCTION

In order to reconstruct the spatialization of the *Interlude*, possible clues can be found in the sonic structure of the piece, in Xenakis' spatialization strategies after the Pavilion, and in Xenakis' aesthetics at the time.

As we are still investigating about the 3-track tape (the final version mentioned by Xenakis was 2'30", while the 3-track tape we found is less than 2 minutes), in the following we take into account (and use for reconstruction) the currently available version by EMF [4], lasting 2'40", mixed down to mono. In his well-known book *Formalized music* [5], Xenakis describes *Concret PH* as an example of "Markovian Stochastic Music" [5]. For the composer, the piece demonstrates that "stochastics is valuable not only in instrumen-

[2] The number of loudspeaker for the route is estimated in 52 by [1].

[3] Letter from the private collection of Peter Wever.

[4] Iannis Xenakis, *Electronic Music*, EMF, CD EM102.

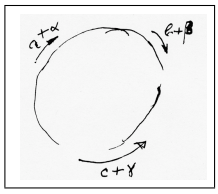


Figure 3. Drawing by Xenakis (unpublished letter to Kalff), explaining the spatialisation of the *Interlude Sonore*.

tal music, but also in electromagnetic music" ([5], p. 43). The same chapter introduces the first theoretical proposal for what would then be called "granular synthesis": *Concret PH* could be considered as its first example [9]. Xenakis, as most scholars, individuates the crucial aspect of the piece in sound material, not in spatialization: that is, in the use of micro, particulated, undifferentiated sound matter to achieving a sense of continuous transformation. *Concret PH* is a textural composition, with no specific temporal macroform, a "cloud filled with splinters of sound" ([9], p. 203), resulting in "dry, but sparkling study" ([16], p. 18). The granular sound matter reveals a specific tactile quality [17], that can remind the concrete surfaces of the Pavilion [18]. While there are many references in literature to Concret PH, still the only detailed analysis is the one proposed by Di Scipio [19]. According to Di Scipio the piece is based on a systemic approach to composition, also evident in *Analogique A et B* [9]. The most relevant aspect of the piece is sound behavior at the micro level, that results from a process of densification, based on the layering of two textures. The "rather simple macroscopic shape" then depends from a single transition from the first to second texture. While Di Scipio's general remarks are very insightful, his analysis is phenomenologically not so convincing [5] . In fact, it is possible to retrieve at least 3 textures (to these, maybe a variation of the second can be added). Figure 4 shows two excerpts from a sonographic representation of *Concret PH* [6] . The piece starts with texture 1, in crescendo: here (Figure 4, top left), the spectrum presents energy spread almost exclusively over 4000 Hz. Even if at 30' some lower impulses can be heard, the second, lower texture enters definitively at 43": new grains are distinctively lower than previous ones, and they appear on the spectrum as black spots in the range between 2500 and 3500 Herz (Figure 4, top, dotdashed box). The double layering of the textures 1 and 2 is kept stable until 84". Then, gliding grains, still from texture 2, appear, with an increasing density and relevance. After 2'05", a third, distinct layer is superimposed: it shows a larger spectrum, both in the low and high frequencies, and it is characterized by a sort of sweeping motion (Figure 4, bottom). The piece ends with texture 3 fading out (2'40"), leaving the texture 1 alone.

Concerning Xenakis' spatializations after the Philips Pavil-

[5] It must be noted that the Di Scipio's analysis was based on a the version of *Concret PH* from the Nonesuch LP, and made use of scarcely readable sonograms produced in 1984.

[6] As the piece is structurally made of large-band impulses, the spectrum is very dense and sonograms are not particularly helpful. Here they are used only to show large discontinuities among textures.

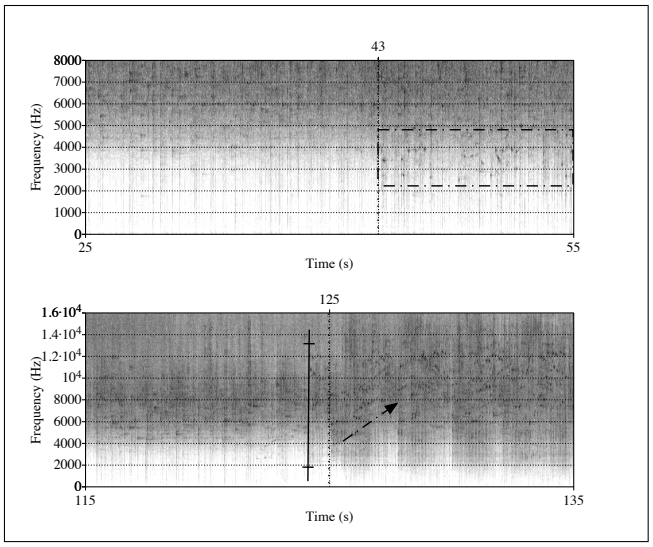


Figure 4. Sonograms of two excerpts from *Concret PH*. Entering of textures 1 (top) and 2 (bottom) (note the different frequency ranges).

ion, the composer made a large use of loudspeakers in many other works. The *Poème électronique* was a model for Xenakis' experimentation with sound and light systems, the *Polytopes* [7] [20] and the electroacoustic work *Bohor* (1968) was for 8 channels. But even if the composer himself has described in depth many of his composition techniques, no detailed info on spatialization strategies in multichannel setup is available, to our knowledge. Still, we know that Xenakis applied to the control of lights and sounds in space the same aesthetic principles and formal procedures he used in his musical composition: as an example, concerning the *Polytope de Montréal* (1967), he stated that his "total experience with musical composition was used to serve light composition" (Xenakis, cit, in [20], 57). As evident in [2], the composer has always strived to a unified form of thought. Hence the need, in order to define a formal model for the control of the sound speed, to take into account some aspects of Xenakis' aesthetics.

Concerning the position of *Interlude/Concret PH* in the Xenakisian corpus, it must be noted that, apart from his seminal work *Metastaseis* (1953-54) (directly inspiring the Pavilion), *Pithoprakta* (1955-56)'s inspiring model came from theory of gas molecules, governing the stochastic distribution of sound events [19], [21]. Analogously, *Achorripsis* (1957-58) can be considered a sonification of probability distributions, following the model of gas diffusion [22] [23]. The same principles are at work in his electroacoustic music. In *Diamorphoses* (1957) he applied stochastic principles, while *Analogiques A et B* introduced Markovian Stochastic Music [5] [19], and the notions of Gabor quanta. Speaking about *Pithoprakta* in 1956, Xenakis underlines the role of the "Gas Parable", one of the "parables" he used to conceptualize the organization of sound events [2]. In the Gas model, the two main parameters to be considered are pressure and temperature: pressure

[7] More, the shape of the *Diatope* (1979) bears a striking resemblance with the Philips Pavilion.

is the density of events, while temperature is the speed of molecules (that is, it represents kinetic energy). Indeed, Xenakis loves to speak about music in terms of its "temperature". This allows to introduce another typical element of Xenakis' geometric imagery: line. *Metastaseis* and the Pavilion are examples of the composer's fascination with ruled surfaces, and glissandos are Xenakis' preferite examples of lines in music. Not by chance, they are put in relation to temperature of molecules [5]. As the sound material of the *Concret PH* is indeed made of sound molecules, the sound spatialization should make them travel around the Pavilion. As in a cloud of particles, masses can travel at different speeds.

In our reconstruction, we embrace a conservative approach, assuming as a plausible hypothesis that Xenakis considered initially three tracks, each one with a different texture: tracks, made of molecular sounds, were not only characterized by specific densities and temperatures of sound molecules (one can think in particular of the gliding grains), but, coherently, also by a specific speed of diffusion in the space. As he was not permitted to use three tracks in the space, these were mixed down onto a single track tape, to be delivered in the route I. As we discussed, it seems that, at the end, the access to the "3D dimension" was possible too, but in some sense Xenakis considered it optional, and available sources are not clear about what was effectively implemented in the Pavilion. The use of the 3rd dimension can be obtained by delivering the other tracks on other sound routes, going up and down in the pavilion, or by diffusing them from fixed positions, most probably from the large clusters above the entrance and exit (hence the reference to "stereophonic" placement by Xenakis). This second hypothesis is consistent with [1]. Following the conservative hypothesis, in our reconstruction the spatialization of the *Interlude* uses a mono track, as it is limited to the horizontal route, in the form of a continuous variation of speed (acceleration/deceleration), with no clusters involved (clusters could be easily added to our design). Also, we assume that the design of the spatialization for other routes could follow the same principle that we are going to discuss. Thus, the final spatialization score would have been made of activation events for route I occurring at variable time intervals (i.e. created by a variable stepping rate).

Taking into account Xenakis' original idea of having three autonomous tracks, from the analysis of *Concret PH*, we assume three different textural layers, where each texture is characterized by a specific speed. In the spatialization design, each texture is given a certain speed, that can be thought as the rotation speed of the layer over the looping horizontal route. Hence, the total rotation speed (the actual parameter to be reconstructed) is the sum of the three different speeds (Figure 5). In order to calculate the speed of each layer, a direct relation between the average duration of the impulsive events and the average rate is assumed: the shorter the event, the lower the average rate. The available speed range, expressed in activation rate for loudspeaker is $[0, 10]$, where 0 represents no motion and 10 loudspeaker/second was the technical limitation for Philips Pavilion hardware, i.e. the maximum rate at which the

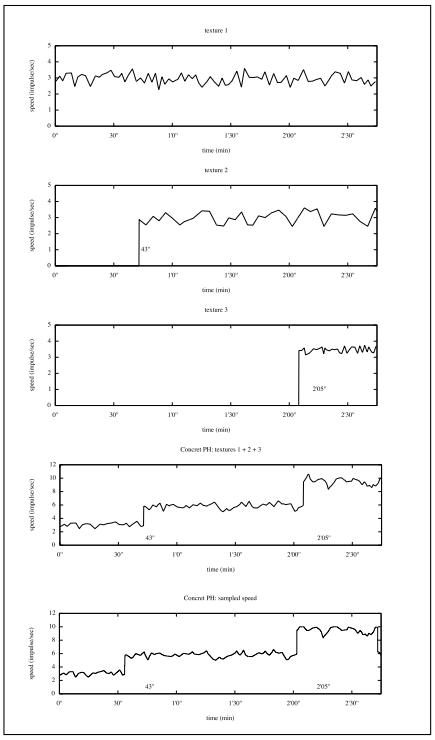


Figure 5. From top to bottom, speeds of layers 1, 2 and 3, overall speed, and sampled overall speed.

loudspeakers can be activated/disactivated [14]. Each texture's speed is defined in a certain interval and actually randomly selected by using a Gaussian distribution (Gaussian distributions were extensively used by Xenakis at that time [5]). As the average speed is proportional to textural density, textures 1 and 2 are opposed to 3, much thicker. The first and the second textures have speeds included in the range $[2, 4]$. Texture 3's speed is in a higher, smaller range, $[3, 4]$. Each texture is given a duration range. During the generation process, a duration is selected in the range, again following a Gaussian distribution. For the duration, speed increases/decreases continuously, as it is calculated by linearly interpolating the starting speed and the next one. In each texture, average duration is proportional to textural density. The duration range in seconds for texture 1 is $[1, 3]$, and $[2, 5]$ for texture 2, as the latter shows a slower pace and the emergence of quasi-pitched grains. Texture 3 is associated the smaller duration range, $[0.3, 2]$, meaning that speed values are selected at the highest rate. This methodology, based on linear interpolation, is based on the relevance of the visual geometric model of the line in Xenakis' work: Xenakis always draws lines on paper while composing, and the ruled surfaces of the Philips Pavilion bear a strict relation to the sketches for the score of *Metastaseis*. Speeds for each texture are represented in Figure 5, with speed expressed in activation impulses per second. In order to define loudspeaker activation/disactivation and to create the final score, we have to remember that speakers are discrete, so it is necessary to define activation/disactivation events by sampling the speed curve. The process works as follow:

1. at t_0 select speed s and activate loudspeaker l_0

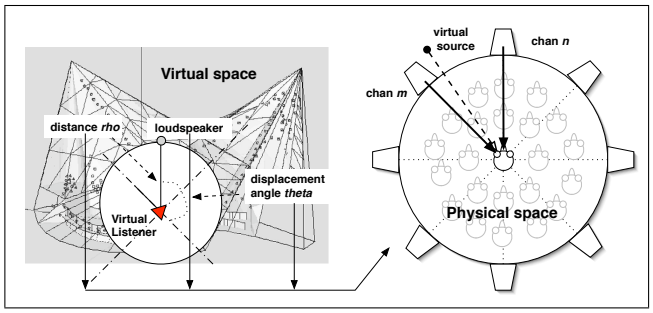


Figure 6. Relations between the virtual space and the final 8-channel setup.

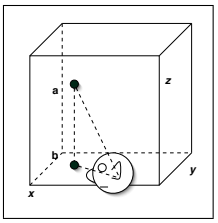


Figure 7. 2D Localization and 3D distance rendering.

2. calculate the duration d of the event, i.e. the time interval to the activation of loudspeaker. As s represents the rate of activation of loudspeaker, if, e.g., $s = 2$, then the next loudspeaker will be activated after $d = \frac{1}{s} = 0.5$ seconds

3. write an event e in the score, with loudspeaker index 0 and duration d

4. wait for d seconds

At time t_0+d, a new speed is selected, and so on, until total duration has passed. As the horizontal route wraps around, the process keeps going on until the total time of the piece has passed. The algorithm can be thought as a Sample and Hold applied to the total speed curve (with variable rate depending on sampled speed). Also, if $s > 10$, then speed is clipped to $s = 10$. The sampled and clipped speed is shown in Figure 5, bottom.

4. IMPLEMENTATION

The proposed *Concret PH*'s spatialization has been implemented in a virtual reality application, that simulates the whole *Poème électronique* inside the Philips Pavilion. By using VR techniques, the resulting installation let the user experience again the original show, placing her/him into a virtual space created by computer graphics and spatialized audio. The application extends the VEP project [1]. Different versions of the VEP application have been implemented by the VEP project: a single-user, immersive setup with headphones and stereoscopic head-mounted display, a "shared" multi-user version with binaural audio (as in the first version) but using screen projection, a non-interactive six-channel version on a DVD-video. Here we present a new version where the audio component is driven by a VR engine, capable of delivering interactive, real-time audio through a multichannel generic setup (Figure 6). Concerning the Philips Pavilion and the *Poème électronique*, it relies on the data provided by the VEP project, and integrates the new findings about *Interlude/Concret PH* discussed before. The general architecture of the real-time versions of the VEP applications features three components: a Computer Graphic Engine, a Control Engine, and an Audio Engine. All the events of the show (concerning the control of audio/video elements of the *Poème électronique*) are recorded sequentially in a score. The score is a text file where each line specifies an event. The Control Engine parses the score in order to retrieve events. At the same time, it also monitors user interaction (i.e. changes of his/her position). For each event, be it a score- or a user one, it sends the opportune commands to the Video and Audio Engines, via OSC messages. Here we do not discuss in details the architecture (see[1]), and we focus instead on the Audio Engine [8]. It uses Vector-Based Amplitude Panning (VBAP) [24], a triangle based generalization of equal-power panning that has been extensively tested with VR applications [25]: it allows to render an arbitrary source in a virtual space on a ring or half dome of loudspeakers (in both cases, the loudspeakers have all the same distances from the ideal listener). The ring/dome can include an arbitrary number of loudspeakers, placed on arbitrary points of the circumference/sphere. In short, by passing the position data related to a virtual source to a VBAP algorithm, the latter can render the virtual position through a physical multichannel setup. In the virtual Philips Pavilion, the sound sources are the 350 loudspeakers, each with a fixed position on the Pavilion's walls. Our final setup made use of an 8-loudspeaker ring, with the audience standing around the ideal listener position, in the centre of the ring (Figure 6). In order to pass from a 3D space (in the Pavilion) to a 2D space (the ring), we discarded the height of the loudspeakers (that is, we projected the loudspeaker positions on the horizontal plane). This means that, with respect to Figure 7, loudspeaker a and b will have the same position. In relation to space simulation, we used VBAP algorithms to recreate sound localization [26]. In order to render distance cues (intensity, direct/reverberat ratio, spectrum) [27], we used the inverse square law to scale amplitude in relation to distance of the loudspeakers from the listener. We avoided to add reverberation (reverberation data for the Pavilion are available from the VEP Project), as it would then be superimposed to the one of the physical space (that cannot be controlled a priori), and we considered spectral filtering not so relevant in the closed space of the Pavilion. While for localization we discard third dimension of original sources, 3D coordinates are still used to calculate distance. As an example, in Figure 7, source b will indeed sound from a position, but at least from b distance. Differently from [1], in this version the virtual listener can move freely in the Pavilion, and her/his position has to be merged with the source position in order to determine the actual orientation to be

[8] In must also be noted that *Concret PH* was to be played during the interlude. As a consequence, no visual elements were present while Xenakis' music was played in the dark Pavilion, thus creating a purely acousmatic context for the piece.

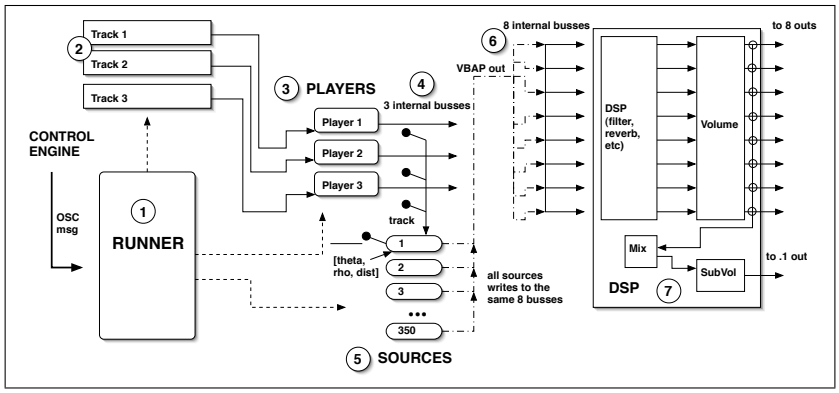


Figure 8. Overall architecture of the Audio Engine.

rendered. The audio application has been developed using the SuperCollider environment: it features a high-level, object-oriented, interactive language together with a real-time, efficient audio server. With respect to other similar projects (Chuck[28], Impromptu [29]), the SuperCollider language combines features common to other general and audio-specific programming languages (e.g. respectively Smalltalk and Csound), and, at the same time, it allows to generate programmatically complex GUIs. It includes native support to OSC protocol, and VBAP algorithms are available from the BEASTmulch library [9] . The architecture of the audio application is shown in Figure 8. It receives OSC messages from the Control Engine. The actual setup includes 8 loudspeakers fed by 8-channel audio, with an added subwoofer receiving a mix down of the 8 audio streams. Its main subcomponents are Runner (1), Players (3), Sources (5) and DSP (7). The Runner handles all the communications with the Control Engine. On initialization, it loads the required audio tracks (2): for each track (3 in case of the *Poème électronique*, 1 for *Concret PH*), it creates a playback unit (3) and routes it to internal busses (4). It creates sources (5) and controls all the processes related to them. In turn, sources represent sources in the pavilion (i.e. loudspeakers), and encapsulate audio DSP capabilities for both playing back audio and spatialization rendering. There are 350 sources, one for each loudspeaker. Each source outputs 8-channel audio via VBAP algorithm, written on 8 common internal busses (6). These busses are routed to the DSP subcomponent (7), that allows for a global control on the audio streams, so that they can be adjusted in relation to the physical listening space. After being scaled through global amplitude scaling ("volume"), the 8 audio streams are mixed down to be sent to the LFE channel. The sub volume provides independent scaling for the subwoofer signal. In real-time interaction, the Control Engine sends event-related messages to the Runner. There are three types of events: startup, score, user. Startup events requires to start/stop the playback units, and are used for syncing the Audio Engine with the Control Engine. Score events are the ones scripted in the score (resulting from reconstructed spatialization), and occurs when a loudspeaker in the Pavilion start or stop receiving audio. In the case that a loudspeaker is started in the virtual pavilion, the Runner calculates the `theta/rho` parameters (representing position in polar coordinates) by merging the position of the virtual source (passed with the message) and the listener's one (stored by the Audio Engine). More, `dist` parameter is calculated, representing the distance of the source from the listener in 3D. Then, the related source is activated, and the required track is routed. The source is passed the spatial parameters (*theta, rho, dist*), so that it can properly process the incoming signal (from the routed track), and render audio. The Runner keeps track of active sources in the "active list", and stores the actual position of the listener. When a loudspeaker is no more active, a stop audio message is sent by the Control Engine: on receiving, the source is paused and removed from the active list. Finally, user events occur when the position of the listener in the virtual space changes, that is when the actual user explores the virtual Pavilion e.g. by means of a joystick. In that case, the listener position is updated, and `theta, rho, dist` parameters are recalculated and passed to sources in the active list. In this way, the 8-channel panning and the scaling for each source are recalculated.

[9] `http://www.beast.bham.ac.uk/research/mulch.shtml`

5. CONCLUSIONS AND FUTURE WORKS

The reconstruction of the *Poème électronique* has shown how many and intermingled are the issues emerging from XX-th century artworks that massively involve short-living and ad hoc technological solutions and devices [30]. These issues, concerning philological, technological, aesthetic aspects, clearly emerge in the case of the *Interlude Sonore*. Our conservative approach has allowed us to re-experience a plausible diffusion of Xenakis' work in a virtual space, and, in the occasion, to develop an innovative technological solution for a multichannel rendering of the Philips Pavilion. The spatialization proposed at the EMF event has received a very positive feedback from the audience and the specialized press:"the digital manipulation of the original sound material was admirable and effective" [31]. In the future, we plan to analyze in depth, and eventually integrate, the three audio tracks of the *Interlude* we found. Also, we are still searching for the original spatialization score by Xenakis. Finally, we plan to develop alternate versions of the spatialization including the third dimension

using the same framework, as VBAP techniques allow to simulate 3D localization.

6. REFERENCES

[1] V. Lombardo, A. Valle, J. Fitch, K. Tazelaar, S. Weinzierl, and W. Borczyk, "A virtual-reality reconstruction of Poème Électronique based on philological research," *Computer Music Journal*, vol. 33, no. 2, pp. 24–47, 2009.

[2] I. Xenakis, *Musique. Architecture*. Paris: Casterman, 1976.

[3] E. Restagno, ed., *Xenakis*. Torino: E.D.T., 1988.

[4] S. Sterken, *Essays on the intersection of music and architecture*, ch. Music ad an Art of Space: Interaction between Music and Architectire in the Work of Iannis Xenakis. Ames, IA: Culicidae, 2007.

[5] I. Xenakis, *Formalized Music. Thought and mathematics in composition*. Stuyvesant, NY: Pendragon, rev. ed. ed., 1991.

[6] M. Treib, *Space Calculated in Seconds: The Philips Pavilion, Le Corbusier, Edgard Varèse*. Princeton, NJ: Princeton University Press, 1996.

[7] E. Schwartz and B. Childs, eds., *Contemporary Composers on Contemporary Music*. Ann Arbor: University of Michigan Press, 1967.

[8] K. Cascone, "The aesthetics of failure: "post-digital" tendencies in contemporary computer music," *Computer Music Journal*, vol. 24, no. 4, pp. 12–18, 2000.

[9] A. D. Scipio, "Systems of embers, dust, and clouds: Observations after xenakis and brün," *Computer Music Journal*, vol. 26, no. 1, pp. 22–32, 2002.

[10] J. Harley, "The electroacoustic music of Iannis Xenakis," *Computer Music Journal*, vol. 26, no. 1, pp. 33–57, 2002.

[11] S. Kanach and C. Lovelace, eds., *Iannis Xenakis. Composer, Architect, Visionary*. New York: The Drawing Center, 2010.

[12] A. Koch, R. Devos, M. van Dijk, S. van Schaik, and P. Wever, *Dichtbij klopt het hart der wereld. Nederland op de Expo 58*. Schiedam: Scriptum Art, 2008.

[13] J. Petit, *Le Corbusier: Le Poème Electronique*. Minuit, 1958.

[14] L. Kalff, W. Tak, and S. L. de Bruin, "The "Electronic Poem" performed in the philips pavilion at the 1958 brussels world fair," *Philips Technical Review*, vol. 20, no. 2–3, pp. 41–53, 1958.

[15] L. Kalff, "Letters." Unpublished collection at Philips Company Archive.

[16] J. Harley, *Xenakis: his life in music*. New York–London: Routledge, 2004.

[17] K. Norman, *Sounding art: eight literary excursions through electronic music*. Burlington, VT: Ashgate, 2004.

[18] M. Fleuret, *Xenakis*, ch. Il teatro di Xenakis, pp. 159–187. Torino: E.D.T., 1988.

[19] A. Di Scipio, "Compositional models in xenakis's electroacoustic music," *Perspectives of New Music*, vol. 36, no. 2, pp. 201–243, 1998.

[20] M. A. Harley, "Music of sound and light: Xenakis's polytopes," *Leonardo*, vol. 31, no. 1, pp. 55–65, 1998.

[21] A. Orcalli, *Fenomenomenologia della musica sperimentale*. Potenza: Sonus, 1993.

[22] E. Childs, "Achorripsis: A sonification of probability distributions," in *Proceedings of the 8th International Conference on Auditory Display (ICAD2002)* (R. Nakatsu and H. Kawahara, eds.), (Kyoto, Japan), 2002.

[23] L. M. Arsenault, "Iannis xenakis's "achorripsis": The matrix game," *Computer Music Journal*, vol. 26, no. 1, pp. 58–72, 2002.

[24] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *JAES*, vol. 45, no. 6, pp. 456–466, 1997.

[25] V. Pulkki and T. Lokki, "Creating auditory displays with multiple loudspeakers using vbap: A case study with diva project," in *Proceedings of the 5th International Conference on Auditory Display (ICAD98)* (S. Brewster, ed.), (University of Glasgow, U.K.), 1998.

[26] J. Blauert, *Spatial Hearing. The Psychophysics of Human Sound Localization*. Cambridge (Mass.)–London: The MIT Press., 1997.

[27] F. Fontana and D. Rocchesso, *The Sounding Object*, ch. Synthesis of distance cues: modeling and a validation, pp. 205–220. Firenze: Edizioni di Mondo Estremo, 2003.

[28] W. Ge and P. Cook, "Chuck: A concurrent, on-the-fly audio programming language," in *Proceedings of the International Computer Music Conference (ICMC)*, (Singapore), 2003.

[29] A. Sorensen, "Impromptu: An interactive programming environment for composition and performance," in *Proceedings of the Australasian Computer Music Conference 2005*, pp. 149–153, ACMA, 2005.

[30] V. Lombardo, A. Valle, F. Nunnari, F. Giordana, and A. Arghinenti, "Archeology of multimedia," in *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, (New York, NY, USA), pp. 269–278, ACM, 2006.

[31] S. Tsigrimanis, "On site: Iannis xenakis. composer, architect, visionary," *The Wire*, p. 79, March 2010.

A LYRICS-MATCHING QBH SYSTEM FOR INTERACTIVE ENVIRONMENTS

Panagiotis Papiotis
Music Technology Group,
Universitat Pompeu Fabra
panos.papiotis@gmail.com

Hendrik Purwins
Music Technology Group,
Universitat Pompeu Fabra
hendrik.purwins@upf.edu

ABSTRACT

Query-by-Humming (QBH) is an increasingly prominent technology that allows users to browse through a song database by singing/humming a part of the song they wish to retrieve. Besides these cases, QBH can also be used to track the performance of a user in applications such as Score Alignment and Real-Time Accompaniment. In this paper we present an online QBH algorithm for audio recordings of singing voice, which uses a multi-similarity measurement approach to pinpoint the location of a query within a musical piece taking into account the pitch contour, phonetic content and RMS energy envelope. Experiments show that our approach can achieve 75.4% Top-1 accuracy in locating an exact melody from the whole song, and 57.8% Top-1 accuracy in locating the phrase that contains the exact lyrics – an improvement of 170% over the basic pitch contour method. Average query duration is 6 seconds while average runtime in MATLAB is 0.8 times the duration of the query.

1. INTRODUCTION

1.1 Presentation of the context

Query by Humming has gained attention as an approach, partly due to the increasing size of music collections; it is far easier to hum/sing the main melody for the song one wants to retrieve than search for it using the title and/or semantic labels. Further signs of the growing presence of QBH as an audio querying concept are also demonstrated by its inclusion as part of the of the MIREX contests since 2006.

However, this is not the only occasion on which QBH can be applied. Real-time Accompaniment systems such as Score Following/Alignment [1] attempt to align an existing score to the performance of a soloist in an online manner, very much like a human accompanist would align and adapt his/her performance to match that of the soloist.

Unfortunately, capable accompanists are hard to come by, especially for pieces of advanced difficulty. For this reason, an intelligent accompaniment system would be very useful for the soloist, as it would allow him/her to train and prepare for a performance without the need of a compromise for a sub-standard accompanist or the reduction of his opportunities to practice for a challenging piece.

In this context, a capable QBH system would be valuable as a front-end; it would automatically locate the soloist's position within the musical piece and provide the starting point for the alignment process. The soloist could stop and start again from a different point in the piece, without having to manually adjust the starting position.

However, virtually every musical piece contains repetitions of the same melody; simply put, there are several starting points for most queried melodies, a detail that is overlooked in most QBH implementations available currently. In the special case of the singing voice, a useful feature that could be utilized to discriminate between identical melodies are the piece's lyrics; with the exception of repeating parts such as the bridge or the chorus, every melodic line is coupled with a different lyric line. Therefore, if one can match the lyrics of the query to the lyrics of the reference, the exact location of the soloist within the piece would be pinpointed with perfect accuracy.

1.2 Related Work

As mentioned before, there is a significant amount of research done in the field of QBH, and as a standalone research field it is increasing in maturity. Early works on content-based audio or music retrieval are primarily based on signal processing and the acoustic similarity of the whole waveform [2].

Recent advances in the field utilize only the pitch contour of the query, which is directly transcribed from audio and compared to MIDI representations of all pieces within a database [3]. This approach yields satisfactory results, but strongly depends on the quality and accuracy of the query. Furthermore, this method performs a certain simplification over the input data to the point where discrimination between two or more candidates becomes a very hard task. Other approaches include a further range of features to calculate similarity, such as rhythm and pitch intervals [4], or relative interval slopes [5].

Predominantly, two different distance metrics are used in order to calculate the similarity between the query and the musical pieces within the database: *frame-based* and *note-based* similarity. Either one has its advantages; frame-base similarity is more accurate and ro-

bust, but is time-consuming. On the other hand, note-based similarity is faster, but offers less precision. A more efficient approach which utilizes the second metric can be found in [6], where the query is transcribed into pitch vectors and a list of candidate melodies is retrieved from the song database using *locality sensitive hashing*.

Another interesting approach from which our work borrows elements is the use of *multi-similarity fusion*, or the combination of the two distance metrics [7]; first, note-based similarity is used in order to quickly filter out the least similar candidates, and then frame-based similarity is applied to more accurately retrieve the best candidates.

Regarding lyrics recognition, a promising approach that is partly similar to our work approach is presented in [8], where a supervised Hidden Markov Model is used to recognize phonemes in a song using a pre-processed lyrics file, with an interesting application in QBH which achieves an accuracy of 57%. Another approach can be seen in [9], where an existing lyrics file is aligned to a vocal performance in Cantonese, using a combination of melody transcription, onset detection and Dynamic Time Warping (DTW) [10].

1.3 Our approach

Since we are trying to locate the exact position of a query within a single musical piece, the conditions and goals are relatively different to most of the cases presented above. Furthermore, the system has to work in a real-time accompaniment context; this restricts the average duration of the queries, since the QBH algorithm has but a small amount of seconds to return the located phrase.

Another goal of this work is to reduce the number of dependencies in terms of input as much as possible; for this reason, we avoided the use of auxiliary MIDI scores for the reference vocals as well as text files containing the lyrics for each phrase. This way, the only prerequisite for this system is a relatively stable audio recording of the reference vocals, such as the vocals in the originally recorded track. This recording is used to match the position of the queries sung by the user; it also serves as a reference through which the user's deviations in time and dynamics can be calculated to align the accompaniment to the user's performance. However, the latter part is still in progress and will not be discussed in this article.

The remainder of this paper is organized as follows: In Section 2, an overview of our system is provided. Section 3 focuses on implementation details for our approach. Section 4 shows our experimental results on the accuracy of the algorithm, and finally Section 5 contains our conclusions and future work recommendations.

2. SYSTEM OVERVIEW

As you can see in Figure 1, our system can be analyzed in four main processing modules. One is the pitch contour post-processing module, and three separate implementations of the Dynamic Time Warping algorithm –

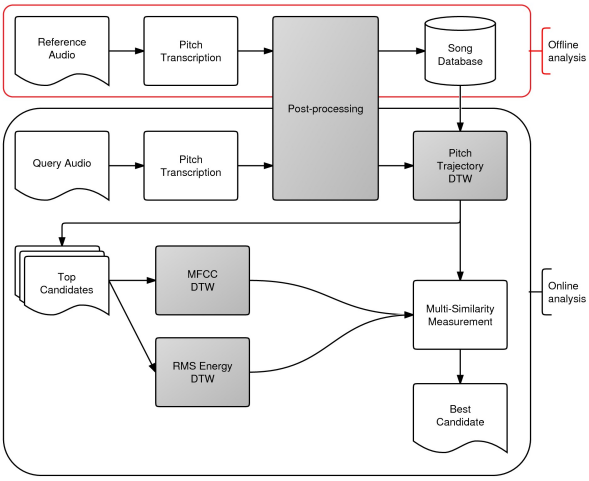


Figure 1. Overview of our system's four modules organized by online and offline analysis.

for the Pitch Contour, Mel-frequency Cepstral Coefficients and RMS Energy respectively. The two basic inputs for the system are the audio recordings of the reference vocals and the query.

2.1 Pitch transcription and post-processing

Both the reference vocals and the query are transcribed with the Yin algorithm [6], which produces a fairly accurate preliminary form of the F0 contour. However, YIN introduces several errors, the most prominent of which are the so-called "octave errors" – falsely choosing a fundamental frequency of twice or half the correct F0.

In order to overcome this problem, we first determine the tonal range of the recording by finding its maximum and minimum value where the aperiodicity of the signal is lower than a given threshold. Knowing the tonal range, we restore the values that are outside it by adding or subtracting a constant number. This way, the contour of the melody remains intact and is just "moved" using a certain offset.

Another problem we had to overcome are points in the recording containing consonants, roughness in the voice, or any brief burst of noise that "pollutes" the melodic content of the recording. Since these points do not have pitch, they can be removed using an aperiodicity threshold over which all values are set to zero. The gaps created are then bridged using an average value.

Finally, to smooth out the curve and speed up the DTW algorithm, we downsample the contour by a factor of 100.

2.2 Contour-based DTW

After obtaining the processed pitch contour of the query and the reference melody, we perform the DTW algorithm between the query and a sliding window of the reference that is equal to the length of the query. This returns a curve with the warping cost for every window of the reference (see Fig.2).

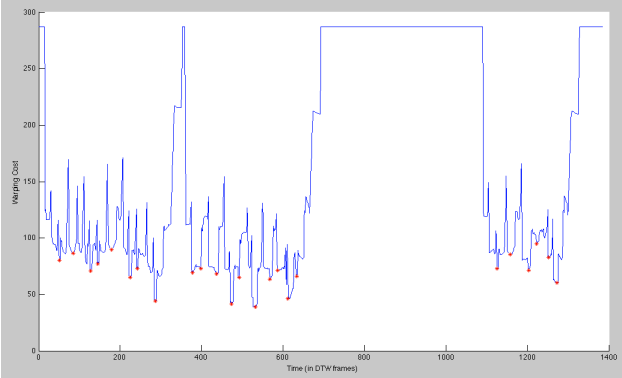


Figure 2. Contour-based DTW costs.

As seen in Figure 2, we select the local minima of this curve as the best candidates for the query. It can be observed that there are at least four phrases with the same melodic contour as the query; this is normal as songs have repeating melodies with different lyrics each time.

2.3 MFCC and RMS energy-based DTW

Having selected the best candidates, we try to match the phonetic content of the query based on the Mel frequency Cepstral coefficients and the pattern of words and silences as represented in the RMS Energy envelope. The DTW algorithm is performed between the retrieved candidates and the query twice more – once for the MFCCs and once for the RMS envelopes, thus producing two sets of warping costs.

2.4 Multi-Similarity Fusion

Finally, all three warping costs are combined, to help determine the best candidate. Each one of the cost vectors is normalized by its norm and added to the final costs vector. The minimum value of this vector is returned, signifying the position where the query was located.

3. IMPLEMENTATION DETAILS

The implementation of the proposed system was done in MATLAB. For the reference melodies, 7 vocal monophonic recordings of songs from the pop/rock genre were used, while the queries were recorded independently by a single user and comprised of 114 phrases with an average duration of 6 seconds each.

The whole system consists of two basic functions: *yinToTrajectory*, which performs the post-processing algorithm described in 2.1, and *QBHLyricFinder,* which calculates the DTW costs as shown in 2.2. In *yinToTrajectory*, the aperiodicity threshold is required as a parameter; values higher than the threshold are removed and replaced with an average value curve. In *QBHLyricFinder*, only the hop-size for the Contour-based DTW has to be adjusted by the user; it has been observed, however, that a hop-size of 400ms or less achieves the best results.

For the pitch contour & RMS energy DTW, Euclidian distance was used in order to construct the similarity matrix. For the Mel-frequency Cepstral coefficients, the cosine similarity between the two MFCC matrices was used, as is also the case with the DTW implementation found in [10].

As we need the system to work as part of an Interactive accompaniment application, computational efficiency is very important: after the user has sung the query, he/she keeps singing; the system must calculate the query's exact position within the reference and start playback from the point the user has reached by that moment. In average, our system's response time is 4.65 seconds for a 6.45-second query. Of course, the response might vary according to the number of candidates chosen during the pitch-contour DTW calculation. Out of these 4.65 seconds, 3.17 correspond to the YIN analysis of the query as well as the post-processing, 1.03 seconds correspond to the Contour-base DTW, and 0.47 seconds to the rest of the algorithm.

4. EXPERIMENTAL RESULTS

As mentioned before, 114 recorded phrases covering the whole of 7 different tracks were used as queries. In our context, a phrase is defined as a small group of words, matched with an individual melody, that stands as a conceptually distinct unit within the song – which is usually a line of the lyrics with its associated melody. Only the best-matching phrase is retrieved by the algorithm; we considered the output of the algorithm a hit, if the phrase returned had an overlap of at least 50% with the query.

Besides the main accuracy of the algorithm, we also calculated for each track the random guess accuracy for lyrics matching, the mean MFCC similarity between the reference and querying voice, the melodic variation of the track and the accuracy of the post-processed pitch contour.

4.1 Random guess accuracy

Since an overlap of at least 50% between the retrieved phrase and the query is considered a hit, the first frame of the retrieved phrase must be located at half the query's length before or after the actual first frame within the reference; more simply put, the overlap between the retrieved phrase and the query can either occur at the query's first or second half, but the duration of the retrieved phrase is always equal in length to the query. Therefore, the range of positions (in number of frames) that are considered correct is equal to

$$f_{query} = \frac{l_{query}}{h}, \quad (1)$$

where l_{query} is the length of the query and h is the hop size of the sliding DTW window, in frames. Similarly, the last frame of the retrieved phrase must be located at half the query's length before or after the actual last frame within the reference, so the range of all possible positions is equal to

$$f_{ref} = \frac{l_{ref} - l_{query}}{h}, \quad (2)$$

where l_{ref} is the length of the reference in frames.

This way, the random guess accuracy can be computed using the following formula:

$$acc = \frac{f_{query}}{f_{ref}} = \frac{l_{query}}{l_{ref} - l_{query}} \quad (3)$$

Some of the songs contain phrases that are repeated throughout its duration, therefore increasing the random guess accuracy. Moreover, when two identical phrases appear sequentially (i.e. the end of the first coincides with the beginning of the second), any frame between the middle of the first repetition and the middle of the second repetition is considered a hit. For these cases, the random guess accuracy is equal to

$$acc_2 = \frac{l_{query} * \left[n_r - (\frac{n_b}{2}) \right]}{l_{ref} - l_{query}}, \quad (4)$$

where n_r is the number of repetitions for that phrase and n_b is the number of shared boundaries between sequential phrases. Since in these cases the accuracy changes according to n_r and n_b, we compute it as a weighted average of all phrases within the song.

It can be argued that two identical phrases, which contain the same lyrics content and melody, might be emphasized differently in each repetition and can therefore qualify as separate phrases; this is currently viewed as a very subtle difference by our approach and such phrases are not treated individually. However, it is a valid case in some types of music (such as in operatic arias) and shall be investigated in the near future.

The overall random guess accuracy for each song as well as the average random accuracy can be seen in Table 1:

Song name	**Random guess accuracy**
She's leaving home	0.076
Butterflies & hurricanes	0.055
Nude	0.036
Bohemian Rhapsody	0.049
A day in the life	0.041
All the small things	0.134
Message in a bottle	0.066
Average baseline accuracy	**0.0471**

Table 1. Individual and average baseline accuracy for lyrics matching.

The average baseline was computed using a weighted sum, according to the number of queries for each song.

4.2 Average accuracy

The average accuracy of our algorithm was calculated as the number of queries located correctly over the total number of queries. In order to evaluate our results clearly and draw conclusions, we also calculated a number of features for each song, which are shown together with the average accuracy in Table 2.

Song ID	**Accuracy**	**Timbre similarity**	**Contour accuracy**	**Melodic variation**
SLH	0.61	0.3505	4	0.23
B&H	0.66	0.4091	4	0.27
N	0.61	0.7506	5	0.61
BR	0.57	0.374	1	0.57
ADITL	0.45	0.6172	2	0.29
ATST	0.6	0.3564	2	0.5
MIAB	0.6	0.2992	2	0.8
Overall	**0.585**			

Table 2. Accuracy and computed features (song IDs are derived from the initials of the song title).

Timbre similarity was calculated as the mean cosine similarity between a query and the relevant phrase from the reference recording, in order to observe how different singers (each with his/her own pronunciation and timbre) influence the lyrics matching.

Pitch contour accuracy was qualitatively graded from 1 to 5, according to the smoothness of the reference pitch contour as well as its similarity with the actual vocal line - it was observed that the pitch contour retains errors and noisy elements even after the post-processing.

Finally, melodic variation was calculated for each song as the number of unique phrases within a song over the total number of phrases in it; a melody is considered unique if its pitch contour is not repeated within the song. High melodic variation characterizes a piece where the vocal melodies are seldom reused, whereas low melodic variation characterizes pieces that feature repetitive melodies.

As our results show, average accuracy for our algorithm is 58.5% with a random guess accuracy of 4.7%, while the accuracy of our program when trying to only locate a phrase with the same contour is 75.4%. We also tested an implementation of the basic QBH algorithm using only the pitch contour to match the queried phrase; the accuracy amounted to 34% when trying to locate a phrase with the same lyrics, and 72.8% when trying to only locate a phrase with the same contour; this demonstrates that using other features besides pitch contour can actually increase the retrieval accuracy even when the objective is not to retrieve phonetic-matching content.

It can be seen from Table 2 that the most evident factor affecting the accuracy is the quality of the pitch transcription (*Contour accuracy*), although the calculation of this feature was qualitative. This is expected, since the performance of the MFCC-based matching is heavily improved when the candidates are fewer and more accurate. Since the timbre of the reference voice is statistically bound to be rather dissimilar to the querying voice, the number of candidates that 'survive' through the Contour-based DTW must be restricted only to the best-matching contours. The melodic variation does not have a big impact on accuracy since, based on our qualitative observations, almost all variations of a queried melody appear among the chosen candidates.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a lyrics-matching Query by Humming algorithm that can be used as a front-end for an interactive real-time accompaniment system specialized for singing voice. An audiovisual demonstration of our system is available on the Internet for evaluation purposes, on the address provided in [11]. Our experiments demonstrate that this approach shows promising results in the context of a time critical, single-output deterministic system. However, our experiments are limited in number and were tested with a single user; efforts to remediate this are currently underway.

An immediate improvement over the algorithm would be a better pitch contour post-processing module for the reference vocal recording, as it is demonstrated that its performance directly influences the accuracy; such an improvement could be the addition of an HMM-based model that would align an existing MIDI score to the reference recording, in order to avoid discontinuities and errors in the reference contour.

Another improvement that would bridge the gap between the contour-matching and the lyrics-matching accuracy would be to utilize the residual part of both the query and the reference recordings, in order to rectify the MFCC-based similarity measure. This way, all melodic content would be discarded when trying to match the lyrics in two phrases with an identical melody.

Finally, as this system is designed to be executed in an interactive environment, a mapping could be gradually performed between the reference recording and the user's voice, in order to increase the lyrics-matching accuracy through extended use.

6. ACKNOWLEDGEMENTS

The second author (HP) holds a Juan de la Cierva scholarship of the Spanish Ministry of Science and Innovation.

7. REFERENCES

[1] A. Cont: "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music", *Proceedings of International Computer Music Conference (ICMC)*, August 2008, Belfast, Ireland.

[2] J. T. Foote: "Content-Based Retrieval of Music and Audio." *In C.-C. J. Kuo et al, editor*, *Multimedia Storage and Archiving System II, Proceedings of SPIE*, volume 3229, pp.138—147,1997.

[3] A. Ghias, J. Logan, D. Chamberlin, B.C. Smith: "Query By Humming - Musical Information Retrieval in An Audio Database", *Proceedings of the third ACM international conference on Multimedia*, pp. 231—236, 1995.

[4] R.J. McNab et al: "Towards the Digital Music Library: Tune Retrieval from Acoustic Input". *Proceedings of Digital Libraries*, pp 11—18, 1996.

[5] A.L.P. Chen, M. Chang and J. Chen: "Query by Music Segments: An Efficient Approach for Song Retrieval". *Proceedings of IEEE International Conference on Multimedia and Expo*, 2000

[6] M. Ryynanen and A. Klapuri: "Query by humming of MIDI and audio using locality sensitive hashing". *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp 2249—2252, Las Vegas, Nevada, USA, Apr. 2008.

[7] L.Wang, S.Huang, S.Hu, J.Liang and B.Xu: "An Effective and Efficient Method for Query by Humming System Based on Multi-Similarity Measurement Fusion", *IEEE International Conference on Audio, Language and Image Processing*, Shanghai, 2008.

[8] A. Mesaros, T. Virtanen: "Automatic Recognition of lyrics in singing", *EURASIP Journal on Audio, Speech, and Music Processing vol. 2010*, Article ID 546047, 11 pages, 2010.

[9] C.H. Wong, W.M. Szeto, K.H. Wong: "Automatic lyrics alignment for Cantonese popular Music", *Multimedia Systems*, vols. 4-5, no. 12, pp. 307—323, 2007.

[10] D. J. Berndt, J. Clifford: "Using dynamic time warping to find patterns in time series", *KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, pp. 359—370, Seattle, 1994.

[11] D. Ellis: "Dynamic Time Warping in MATLAB", *http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/*, retrieved on April 2010.

[12] http://www.youtube.com/user/qbhlyricsalignment

ANALYZING LEFT HAND FINGERING IN GUITAR PLAYING

Enric Guaus, Josep Llus Arcos
Artificial Intelligence Research Institute, IIIA.
Spanish National Research Council, CSIC.
{eguaus,arcos}@iiia.csic.es

ABSTRACT

In this paper, we present our research on left hand gesture acquisition and analysis in guitar performances. The main goal of our research is the study of expressiveness. Here, we focus on a detection model for the left hand fingering based on gesture information. We use a capacitive sensora to capture fingering positions and we look for a prototypical description of the most common fingering positions in guitar playing. We report the performed experiments and study the obtained results proposing the use of classification techniques to automatically determine the finger positions.

1. INTRODUCTION

Guitar is one of the most popular instruments in western culture. The guitar (and the music it produces) has been object of study in many disciplines, i.e. musicology, sociology, physics or computer science. Focusing on acoustic and signal processing disciplines, there are many interesting studies explaining its physical behavior and produced sound [1, 2]. Nevertheless, the essence of guitar music is sometimes reflected by subtile particularities which are completely dependent on the players, styles, and musical genres. Although some successful approaches exist in the literature [3], these particularities are, sometimes, difficult to identify only with recorded audio data. The richness of the guitar expressivity raises a challenge that, even analyzing each string individually, i.e. using hexaphonic pickups, it is still partially tackled. In this context, caption of gestures in guitar performances becomes a good complement to the audio recording.

The study of performer gestures in music is not new. For instance, Young [4] presented a system to capture the performance parameters in violin playing. Focusing on the guitar, there are some interesting approaches studying the gestures of guitar players [5, 6]. Centering on the finger movements, the available approaches are traditionally based on the analysis of images. Burns and Wanderley [7, 8] proposed a method to visually detect and recognize fingering gestures of the left hand of a guitarist. Heijink and Meulenbroek [9] proposed the use of a three-dimensional motion tracking system (Optotrak 3020) to analyze the behavior of the left hand in a classical guitar. Norton [10] proposed the use of another optical motion caption system based on the Phase Space Inc., with quite successful results. Although these optical systems have proved to partly solve and represent guitar gestures, some occlusion problems may appear in specific finger positions. The proposed acquisition system is a good complement to the existing ones.

Our research focuses on understanding of particular articulations used by different players, styles or musical genres. For that, we need to capture gesture information from the left hand and to detect its exact position. With such information, we can (1) detect the fingering in a given score, and (2) predict the possible articulations and plucked strings even before the sound is produced. The goal of this paper is to present a model that detects the left hand position, based on gesture information, using classification techniques.

The paper is organized as follows: First, in Section 2, we describe the sensors we use. Then, Section 3 shows the list of recorded excerpts, and explains the pattern creation process from the recorded data. Next, in Section 4, we carefully analyze the obtained recordings, propose the use of classification techniques to automatically classify the patterns, and analyze the results. Finally, we summarize the results achieved, present research conclusions, and propose the next steps of our research in Section 5.

2. ACQUISITION

The acquisition system is based on capacitive sensors, described in [11]. Capacitive sensors are not new. In 1919, Lev Termen invented the Theremin, considered the first electronic instrument in history. Lev Termen exploited the capacitive effect of a player near two antennas, one controlling the pitch and the other controlling the loudness, of an harmonic signal. More recently, new musical interfaces also use capacitive sensors to control musical parameters [12, 13].

The proposed system consists of an array of capacitive sensors, mounted on the fretboard of the guitar, configured in *load mode* [14], where the distance between the electrode and a single object (the performer's finger in our case) is measured through a change in capacitance of the electrode to ground. These sensors provide information relative to the presence of fingers into that specific fret. Moreover, depending on the number of fingers present in a given fret, the position of these fingers, and the pressure of the fingers to the strings, the response of the sensors differ.

Figure 1. Gesture caption system based on capacitive sensors (mounted on the fretboard) and Arduino (mounted on the body).

Capacitive variations are collected by Arduino [1], an open-source electronics prototyping platform, programmed using Capsense [2], a capacitive sensing library for Arduino. Capsense converts the Arduino digital pins into capacitive sensors that are used to sense the electrical capacitance of the human body. The acquisition system is shown in Figure 1.

As reported in [11], capacitive sensors can be noisy, and crosstalk between measured capacitances at different frets may appear. Moreover, the finger position in a given left hand situation is never exactly the same, depending on musical parameters (loudness, style, etc.) or the player (length of the fingers, etc.). Because of these two reasons, collected data can not be directly processed, and we propose the use of automatic classification techniques to tackle the problem.

3. RECORDINGS

3.1 Score

In usual guitar playing conditions, the index finger over a fret (not necessarily pressing) defines a position. The following fingers are, by default, on the three following frets. Then, the score defines the exceptions to this default fingering. Beyond that, although the score does not modify this default fingering, the fingers can press different strings. So, we have to address our problem in two dimensions: (1) the overall hand position, defined by the index finger, and (2) the played strings at that position, from 1 (high pitch string) to 6 (low pitch string). The huge number of possible finger combinations forces us to organize them according to a given criterion. The parameters we can play with are:

Hand position: The hand can move up and down the fretboard. In our case, the number of sensorized frets is 10, which allows us to move the hand from fret 1 (with fingers over frets 1, 2, 3, and 4) to fret 7 (with fingers over frets 7, 8, 9 and 10), using the default fingering.

Finger positions: Each fret can be excited by a different number of fingers. We consider there are 5 possible situations: 0, 1, 2, 3 and 6 (bar). ”0” means that the fret is not active (i.e. there is no finger acting on that fret), ”1” means that only one finger is acting on that fret, whatever the string is pressing, and so on. A 6 (bar) means that the full index finger is acting on that fret all over the strings. We also made some recordings with a half-bar (pressing only strings 1, 2 and 3), but for this study, we consider half-bars as normal bars.

Pressed strings: For each finger position and default fingering, there are multiple combinations for pressing strings, as shown in Table 1. From all the available combinations, there are some which are not really used because of (a) the hand can not physically hold that combination, or (b) they have no musical meaning. The highlighted combinations in Table 1 represent the recorded cases.

Beyond that, it is important to distinguish between positions that can seem similar (i.e. *1000* and *0010*) but the hand position is completely different and, as a consequence of that, the residual capacitive measure from the other fingers is different. The use of one of these two options is determined by the musical context, which is not covered in this paper. Then, for simplicity, we will skip these alternative recordings.

1st. finger	1 finger/fret	2 fingers/fret	3 fingers/fret
bar	**6000**	6200	6300
	6001	6201	6030
	6010	**6210**	6003
	6011	**6020**	
	6100	6021	
	6101	**6120**	
	6110		
	6111		
1 finger	**1000**	**1200**	1300
	1001	1201	1030
	1010	1210	1003
	1011	1020	
	1100	1021	
	1101	1120	
	1110		
	1111		
2 fingers		**2000**	
		2100	
		2200	

Table 1. Finger activation combinations for each default fingering position. 4 digits refer to 4 successive frets. Each digit corresponds to the number of fingers pressing at the same fret. These positions can be played in different hand positions and in different strings. *6* refers to bar activation, *1* refers to 1 finger activation at any string, *2* refers to 2 finger activation at the same fret at any strings, and *3* refers to 3 finger activation at the same fret at any strings. The highlighted combinations represent the recorded cases.

[1] www.arduino.cc

[2] http://www.arduino.cc/playground/Main/CapSense

Position	Played strings	Category
1000	s1, s2, s3, s4, s5, s6	1000a
1010	s5s6, s4s5, s3s4, s2s3, s1s2	1010a
1010	s4s6, s3s5, s2s4, s1s3	1010b
1010	s3s6, s2s5, s1s4	1010c
1100	s5s6, s4s5, s3s4, s2s3, s1s2	1100a
1100	s4s6, s3s5, s2s4, s1s3	1100b
1100	s3s6, s2s5, s1s4	1100c
1110	s5s4s6, s4s3s5, s3s2s4, s2s1s3	1110a
1110	s4s5s6, s3s4s5, s2s3s4, s1s2s3	1110b
1200	s5s6s4, s4s5s3, s3s4s2, s2s3s1	1200a
1200	s4s6s5, s3s5s4, s2s4s3, s1s3s2	1200b
2000	s6s5, s5s4, s4s3, s3s2, s2s1	2000a
2100	s6s4s5, s5s3s4, s4s2s3, s3s1s2	2100a
2100	s5s4s6, s4s3s5, s3s2s4, s2s1s3	2100b
2200	s5s3s4s2, s4s2s3s1	2200a
6000	full, half	6000a
6010	s5, s2	6010a
6020	s5s4, s4s3	6020a
6100	s5, s2	6100a
6110	s3s5, s2s4	6110a
6120	s3s5s4, s2s4s3	6120a
6210	s4s3s5, s3s2s4	6210a

Table 2. Detailed list of all the recorded positions, specifying the played strings. Each recording includes the hand position moving from fret 1 to 7. The s1..s6 stands for the played string. Each string specification follows an ascending order from finger 1 to 4. In this paper, we refer these positions according to the *Category* column.

The recorded positions are not all the complete combinations. The recorded subset represents, under our point of view, the most common situations in real guitar performances, and also covers some specific situations in which the position recognition presents a difficulty (i.e. *6020* vs *6120*). For each of the proposed positions, several string combinations have been recorded. The same configuration of fingers over the frets also include different possibilities. For instance, the position *1200* may represent *Am* with fingers 1,2, and 3 at strings 2, 4 and 3, respectively, or the *Emaj* with the same fingers at strings 3, 5, and 4, respectively, by moving the whole hand 1 string down. In our analysis, we consider these positions are equivalent. Beyond that, the same position *1200* may represent *Am* with fingers 1,2, and 3 at strings 2, 4 and 3, respectively, or *D7* with fingers 1, 2, and 3 at strings 2, 3 and 1, respectively. Note how the order of the fingers has changed. In our analysis, we study whether these positions present an equivalent response or not.

Table 2 shows a detailed list of all the recorded positions, specifying the played strings. Each recording includes the hand position moving from fret 1 to 7. From the multiple options for each configuration, we have used that one covering the worst case, i.e., we recorded *6110s3s5* instead of *6110s5s3* because, in the first case, the hand is near the fretboard producing a higher crosstalk between the

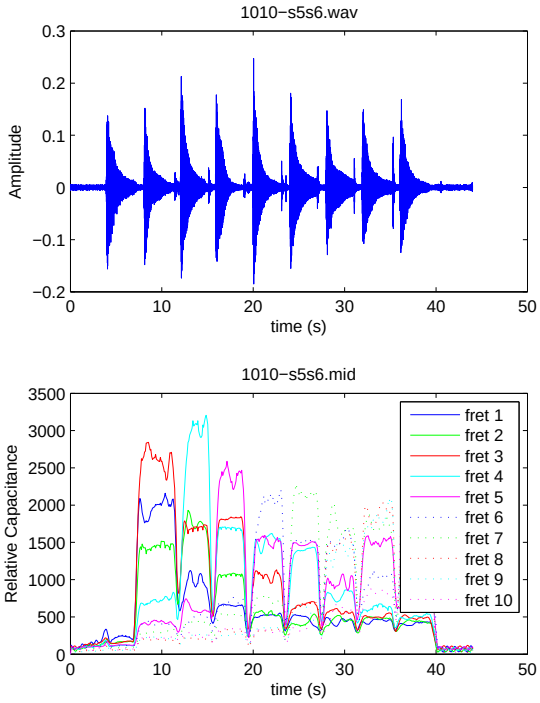


Figure 2. recorded audio and data from capacitive sensors for the 1010-s6s5 position.

measured data from frets.

3.2 Data processing

For all the recordings detailed in Table 2, the audio data from a microphone, and the data from capacitive sensors is captured. Data from capacitive sensors is converted to MIDI. We use 10 MIDI channels, one for each fret, and the information is stored as PitchBend messages to obtain a better resolution. As explained in [11], MIDI data provided by the Arduino does not have a constant sampling rate. We apply automatic resampling obtaining a constant sampling rate sr=30[Hz], which is quite low but accomplishes our requirements. Each hand position has a duration of 4 beats in a 4/4 bar at 60[bpm]. The first bar is used as pre-roll, the second bar is used to play an open strings position in all the recordings, and the specified position starts at the 3rd. bar. Figure 2 shows an example of the recorded audio and gestural information. All the recorded MIDI files can be downloaded at *www.iiia.csic.es/guitarLab/*.

The goal of this paper is to obtain models for each fingering position. We assume the collected data for the same position played at different hand positions is similar (that is, from frets 1 to 7). Then, we collapse all the information for each recording (moving the hand from fret 1 to 7) and build a pattern for that finger position. In order to avoid possible variations produced by the hand movements, we only use information from beats 2 and 3 of every bar, in which we assume the hand position is stable, and compute the mean for all the acquired data from sensors in this period of time. We know the extracted information from bars

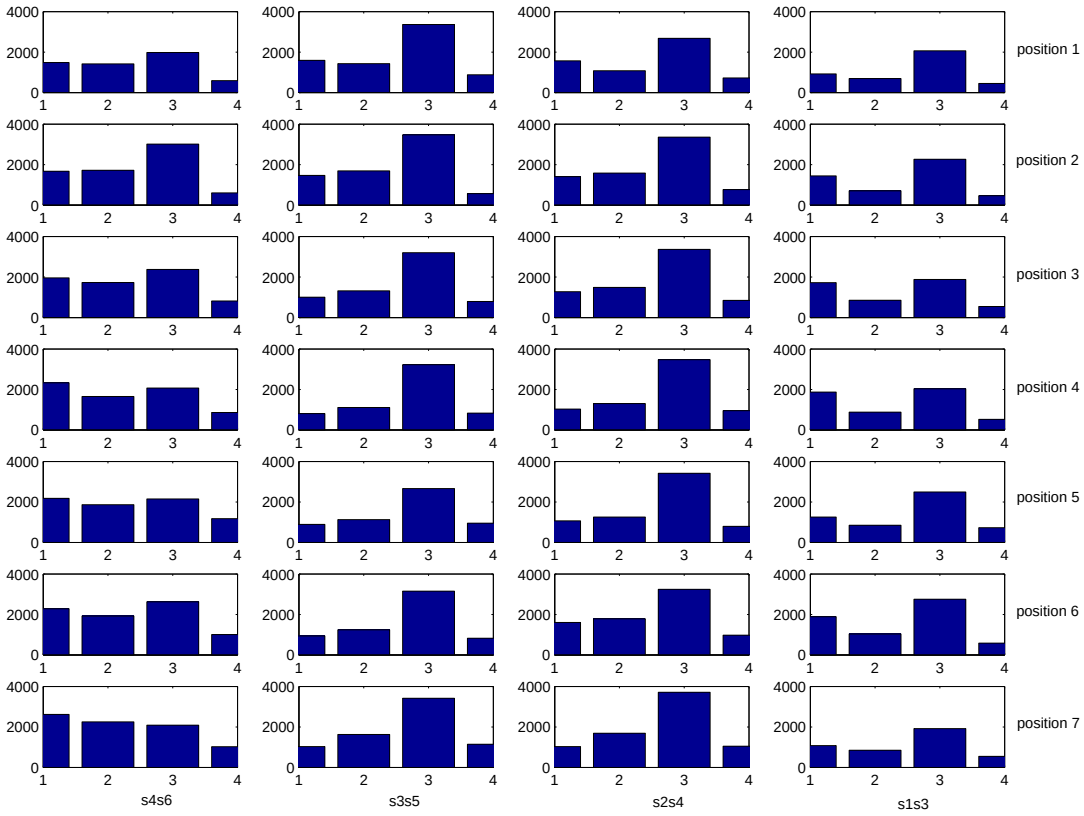


Figure 3. Patterns for finger positions 1, 2, 3 and 4 with respect to the fret position of the index finger, collected for the *1010b* position. Each column corresponds to the same pattern modifying reference frets (from 1 to 7), and each row corresponds to the same pattern modifying strings (s4s6, s3s5, s2s4, and s1s3). Vertical scale refers to measured capacitance.

2 and 3 may differ from the information obtained in other scores (we are a bit conservative, here) but our goal is to obtain the patterns in which the real and faster recordings will be compared to. These means are used to build a pattern for finger positions 1, 2, 3 and 4 with respect to the fret position of the index finger. After some preliminary experiments, we may assume that the information from the other frets is not relevant. These patterns are also collapsed through playing the same finger position at different strings.

In summary, we create, for each position, a pattern for frets 1 to 4 (relative to the position of the index finger) moving the position horizontally on the fretboard (moving the hand from low pitches to high pitches) and vertically (moving the strings from low pitch to high pitch). Figure 3 shows an example of some individual patterns collected to create the *1010* position. Plottings for all the patterns can be downloaded at *www.iiia.csic.es/guitarLab/*.

4. ANALYSIS

In this section, we present the patterns that define different finger positions and the analysis of collected data. Specifically, we are interested in verifying the following hypotheses: (H1) Moving up and down the same position through strings does not change the pattern; (H2) Moving up and down the same position through the fretboard does not change the pattern; (H3) The presence of a bar is always detected and it does not mask the information of following frets; (H4) Positions with one finger per fret can be detected; (H5) Positions with more than one finger per fret can be detected; and (H6) Different finger positions under the same fret configuration present a different the pattern.

The analysis of the collected data is divided in three parts. First, we describe how patterns are created. Then, we analyze whether the obtained patterns are coherent with what we expected. Finally, we analyze whether the obtained patterns can discriminate between different positions automatically.

4.1 Pattern creation

For all the obtained patterns (some of them are shown in Figure 3) and for each recorded position, the behavior is similar. This means that the given values and slopes are equivalent for each row, that is, the same pattern is obtained by playing at different reference frets by moving the hand horizontally on the fretboard, and for each column,

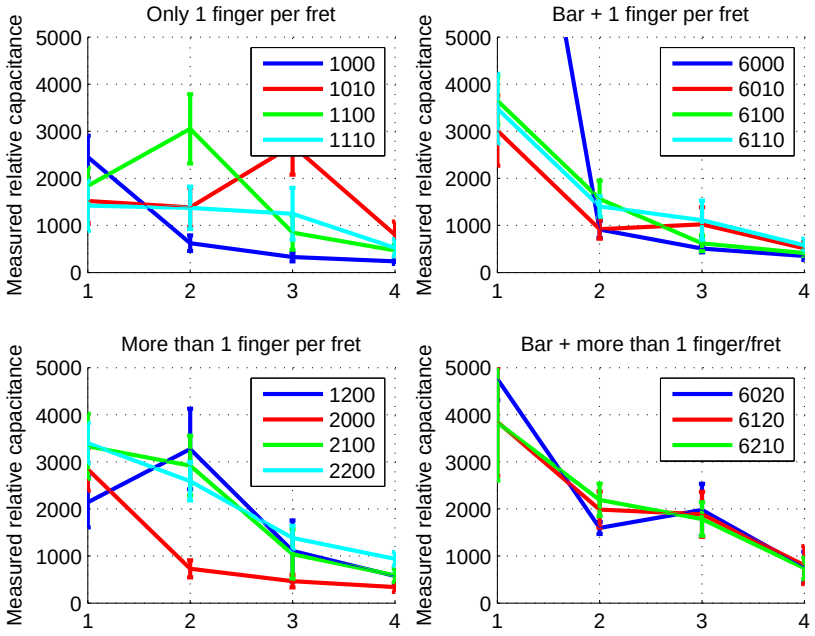


Figure 4. Patterns obtained from means and standard deviations for all the recordings at different finger positions. Position *6000* jumps to 16000, but we skip the vertical scale from 0 to 5000 to ease visual comparison.

that is, the same pattern is obtained by playing at different strings. This result verifies the hypotheses H1 and H2. Then, we may group the patterns for each detailed position in Table 2 into one of the the 22 proposed categories.

4.2 Study of patterns

As expected, the patterns captured with capacitative sensors are not linear combinations of the basic *1000*, *0100*, *0010*, and *0001* patterns. That is, the finger positions are influencing neighboring frets. However, the slopes are consistent with the activated frets. For instance, *6210* recordings present a descending slope whereas *6120* recordings tend to emphasize a sub-peak at third fret.

Regarding finger combinations with a bar, the experiments demonstrate that the presence of a bar does not mask the other fingers (see Figure 4). Indeed, the presence of a bar generates more stable positions (diminishing standard deviation). This result verifies hypothesis H3. Positions *1000* and *2000* can be confused because the slope is similar and the unique difference is the absolute value of the first fret. Although this value is higher at position *2000*, the difference is not large enough to establish a decision point.

Finger combinations in which consecutive frets are activated, present a more clear behavior, both in terms of slope and small deviation. The clearest exponents are recordings with only one finger (*1000*) or a bar (*6000*) pressing the strings, but positions like *1110*, *1200*, *2000*, *2100*, and *2200* follow also a clear behavior.

Two finger combinations require a deeper analysis: *1100* and *1010* (see Figure 4). The two combinations were played with the second active finger pressing lower strings, and lower capacitive values were expected. But higher values were obtained. Regarding position *1100*, the measured relative capacitance is really similar to position *1200*, thus, our system won't be able to distinguish among these to finger combinations. Regarding position *1010*, the first finger sometimes causes a low activation (see Figure 3). Moreover, because the middle finger tends to be close to the fretboard, the measured relative capacitance in the second fret is similar to the measured when one finger is present. These observations partly verify hypotheses H4 and H5, and the use of an automatic classification algorithm will help us to study them in detail.

4.3 Automatic detection

Once we have verified the measured patterns mostly agree with the expected ones, we analyzed whether an automatic classifier might identify them. We have 22 categories (including 75 possible finger combinations) recorded at 7 reference fret positions, that is, a data-set with 525 recordings. As discussed in Section 4.2, not only the absolute values are important in the analysis, but the slopes. In order to include slope relative information to the system, we computed the difference of the means from one fret with respect to the previous one.

The baseline for random classification is 1/22=4,54%. For simplicity, we use a K-nearest neighbours classifier (with K=3) and evaluate using 10-fold cross validation. Results provide an overall accuracy of 44,6% (weighted averaged precision = 0.449, weighted averaged recall = 0.446, weighted averaged f-measure = 0.435). The confusion matrix is shown in Figure 5, and precision and recall values for individual categories are shown in Table 3.

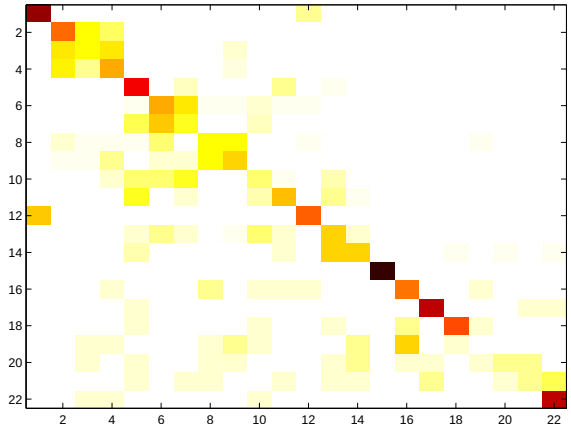


Figure 5. Results of automatic classification using K-nearest neighbours with K=3. Rows indicate categories that should be classified and columns indicate automatically classified categories. Indexes follow these categories: (1)1000a, (2) 1010a, (3) 1010b, (4) 1010c, (5) 1100a, (6) 1100b, (7) 1100c, (8) 1110a, (9) 1110b, (10) 1200a, (11) 1200b, (12) 2000a, (13) 2100a, (14) 2100a, (15) 2200a, (16) 6000a, (17) 6010a, (18) 6020a, (19) 6100a, (20) 6110a, (21) 6120a, and (22) 6210a.

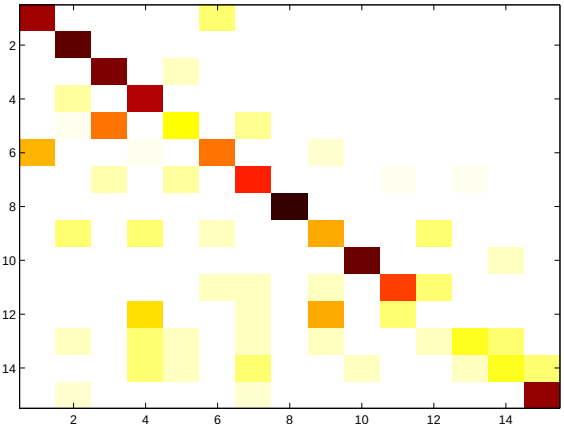


Figure 6. Results of automatic for collapsed categories classification using K-nearest neighbours with K=3. Rows indicate categories that should be classified and columns indicate automatically classified categories. Indexes follow these categories: (1)1000, (2) 1010, (3) 1100, (4) 1110, (5) 1200, (6) 2000, (7) 2100, (8) 2200, (9) 6000, (10) 6010, (11) 6020, (12) 6100, (13) 6110, (14) 6120, and (15) 6210.

Rows indicate categories that should be classified and columns indicate automatically classified categories

Position 16 (*6000*) is perfectly classified. But we observe some confusions between the other positions. First, indexes 1 and 12 (positions *1000* and *2000*, respectively) are the worst classified, but confusions are only among them. This is an expected confusion and it does not affect the identification from the different positions at all. Beyond that, we also observe important confusions between indexes 2, 3, and 4, which correspond to positions 1010a, 1010b, and 1010c, respectively. Note how all these confusions belong to position *1010*, but changing the finger's order, that is, they are equivalent. The number of fingers pressing the frets is the same, and our sensor is not designed to distinguish between them. In a similar way, more confusions can be found between indexes 5, 6, and 7 (positions *1100a*, *1100b*, and *1100c*, respectively), between indexes 8 and 9 (positions *1110a* and *1110b*, respectively), between indexes 10 and 11 (positions *1200a* and *1200b*, respectively), and between indexes 13 and 14 (positions *2100a* and *2100b* respectively). But the number of fingers over the frets is always the same. In the forthcoming positions, with the presence of a bar, confusions are more spread in the space, because the number of fingers on the frets is maximum.

We repeat the automatic classification process by collapsing the equivalent positions (see Table 2). With the resulting 15 categories, and the baseline for random classification is 1/15=6.67%, We achieved an overall accuracy of 69.5% (weighted averaged precision = 0.67, weighted averaged recall = 0.695, weighted averaged f-measure = 0.673). The confusion matrix is shown in Figure 6, and precision and recall values for individual categories are shown in Table 3. Only significant two confusions are still remaining: between positions *1000* and *2000*, and between positions *1100* and 1200, as reported in Section 4.2. For the other finger combinations, confusions are not significative and more spread in the space. Thus, the behavior of the automatic classifier is coherent. To conclude, hypotheses H4 and H5 are partially verified, and hypothesis H6 is verified.

5. CONCLUSIONS

The overall goal of our research is to understand expressivity in guitar performances through particular articulations used by different players, styles or musical genres. For that, we need to capture gesture information from the left hand to analyze the fingering and possible articulations.In this context, this paper presented a model that detects the left hand position, based on gesture information, using classification techniques.

We proposed an acquisition system based on capacitive sensors, we discussed the scores and formats for recordings and analyzed the results directly from the data and using a state of the art automatic classifier. We proposed a list of hypotheses that were practically verified, but results using the proposed automatic classifier can be improved. For that, more research is required. Specifically, we will focus our efforts on improving the gesture acquisition system, by including information from hexaphonic pickup, and musical context information to the classification algorithm.

6. ACKNOWLEDGMENTS

This work was partially funded by NEXT-CBR (TIN2009-13692-C03-01), IL4LTS (CSIC-200450E557) and by the Generalitat de Catalunya under the grant 2009-SGR-1434.

Position	Expanded		Collapsed	
	Precision	Recall	Precision	Recall
1000	0.735	0.857	0.714	0.833
1010s3s6	0.324	0.524	0.85	0.944
1010 s4s6	0.333	0.286		
1010 s5s6	0.395	0.429		
1100 s3s6	0.333	0.714	0.681	0.889
1100 s4s6	0.286	0.429		
1100 s5s6	0.250	0.257		
1110 s4s5s6	0.364	0.286	0.722	0.813
1110 s5s4s6	0.357	0.357		
1200s4s6s5	0.227	0.179	0.483	0.292
1200 s5s6s4	0.478	0.393		
2000 s6s5	0.655	0.543	0.625	0.500
2100 s5s4s6	0.323	0.357	0.689	0.646
2100 s6s4s5	0.556	0.357		
2200s4s2s3s1	0.688	0.786	0.750	0.857
6000	1.000	1.000	1.000	1.000
6010	0.438	0.50	0.333	0.417
6020 s5s4	0.786	0.786	0.917	0.917
6100	0.727	0.571	0.636	0.583
6110	0.000	0.000	0.00	0.000
6120s3s5s4	0.500	0.143	0.500	0.250
6210 s4s3s5	0.400	0.143	0.500	0.250

Table 3. Precision and recall for automatic classification for (a) all the fingering positions individually classified (See Figure 5), and (b) collapsed fingering positions (See Figure 6).

7. REFERENCES

[1] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer-Verlag, New York, NY, 1988.

[2] B. E. Richardson, "Classical guitar construction: The acoustician's tale," *The Journal of the Acoustical Society of America*, vol. 117, p. 2589, April 2005.

[3] C. Erkut, V. Valimaki, M. Karjalainen, and M. Laurson, "Extraction of physical and expressive parameters for model-based sound synthesis of the classical guitar," in *108th AES Convention*, pp. 19–22, February 2000.

[4] D. Young, "The hyperbow controller: Real-time dynamics measurement of violin performance," in *Proc. of New Interfaces for Musical Expression*, 2002.

[5] M. Laurson, C. Erkut, V. Välimäki, and M. Kuuskankare, "Methods for modeling realistic playing in acoustic guitar synthesis," *Comput. Music J.*, vol. 25, no. 3, pp. 38–49, 2001.

[6] M. M. Wanderley and P. Depalle, "Gestural control of sound synthesis," *Proc. IEEE*, vol. 92, pp. 632–644, April 2004.

[7] A. Burns and M. Wanderley, "Computer vision method for guitarist fingering retrieval," in *SMC'06: Proceedings of the Sound and Music Computing*, May 2006.

[8] A. Burns and M. Wanderley, "Visual methods for the retrieval of guitarist fingering," in *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*, pp. 196–199, June 2006.

[9] H. Heijink and R. G. J. Meulenbroek, "On the complexity of classical guitar playing:functional adaptations to task constraints," *Journal of Motor Behavior*, vol. 34, no. 4, pp. 339–351, 2002.

[10] J. Norton, *Motion capture to build a foundation for a computer-controlled instrument by study of classical guitar performance*. PhD thesis, Stanford University, September 2008.

[11] A. AAA, B. BBB, C. CCC, and D. DDD, "Deleted for anonymous submission," in *Deleted for anonymous submission*, pp. 1–2, 2011.

[12] J. Paradiso and N. Gershenfeld, "Musical applications of electric field sensing," *Computer Music Journal*, vol. 21, no. 2, pp. 69–89, 1997.

[13] S. Hughes, C. Cannon, and S. Modhráin, "Epipe : A novel electronic woodwind controller," in *Proc. of New Interfaces for Musical Expression*, pp. 199–200, 2004.

[14] E. Miranda and M. Wanderley, *New Digital Musical Instruments: Control And Interaction Beyond the Keyboard (Computer Music and Digital Audio Series)*. A-R Editions, Inc., 1st ed., 2006.

VOICE CONVERSION: A CRITICAL SURVEY

Anderson F. Machado*
Computer Science Dept. – University of São Paulo
dandy@ime.usp.br

Marcelo Queiroz*
Computer Science Dept. – University of São Paulo
mqz@ime.usp.br

ABSTRACT

Voice conversion is an emergent problem in voice and speech processing with increasing commercial interest, due to applications such as *Speech-to-Speech Translation (SST)* and personalized *Text-To-Speech (TTS)* systems. A *Voice Conversion* system should allow the mapping of acoustical features of sentences pronounced by a source speaker to values corresponding to the voice of a target speaker, in such a way that the processed output is perceived as a sentence uttered by the target speaker. In the last two decades the number of scientific contributions to the voice conversion problem has grown considerably, and a solid overview of the historical process as well as of the proposed techniques is indispensable for those willing to contribute to the field. The goal of this text is to provide a critical survey that combines historical presentation to technical discussion while pointing out advantages and drawbacks of each technique, and to bring a discussion of future directions, specially referring to the development of a perceptual benchmark process for voice conversion systems.

1. INTRODUCTION

Speech is an inherently human communication tool. The development of computational systems that process speech in various ways is a very interesting and important challenge. Systems that concentrate on the intelligible content of speech, such as speech recognition and text-to-speech systems, have received widespread attention due to important applications in providing accessibility for disabled individuals, as well as applications in human-computer interface design and in security systems. Some systems focus mainly on the timbral quality of speech, for instance speaker identification systems, whereas others are equally concerned about intelligible and timbral aspects, such as singing voice synthesis [1].

This paper concentrates on the *Voice Conversion (VC)* problem as introduced by Childers et al. [2], which is the task of converting a sentence uttered by a source speaker in such a way that the converted result appears to be the same sentence spoken with a different voice, i.e. that of a target speaker. It is important to make a few distinctions between (VC) and related problems clear.

**Acknowledgements: The authors would like to thank FAPESP and CAPES.*

Voice Transformation is a general problem that encompasses all tasks and methods that modify any features of a voice signal; for instance, pitch shifting or time-stretching a recorded sentence are examples of voice transformations. *Voice Morphing* is a term borrowed from image processing, and is the special case of voice transformation where two voices are blended to form a virtual third voice, where usually the two source voices speak or sing the same thing synchronously. Instances of these techniques have been made known to the general public through films such as *Farinelli* (where a soprano and a countertenor voices are blended to make up a pretended castratto voice) or *Alvin and the Chipmunks* (where chipmunks voices are pitch-shifted/formant-corrected actors voices).

A *Voice Conversion* system takes into account both the *timbre* and the *prosody* of the source and target speakers. While timbre and prosody are qualities that are easy to recognize and hard to define in general terms, in the specific context of voice conversion timbral features are usually associated with the dynamic spectral envelope of the voice signal, whereas prosody is related to pitch/energy contours and rhythmic distribution of phonemes.

In order to define the transformations related to timbre and prosody, VC systems usually depend on a training phase, which may be **text-dependent** or **text-independent**. In the first case, both source and target speakers have to record the same sentence; after that, both recordings are time-aligned (using for instance *Dynamic Time Warping* [3, 4, 5]), and acoustic features are mapped synchronously between recordings.

In the *text-independent* case [6], source and target speakers are not required to record the exact same sentences. Recordings are usually segmented into frames which are mapped into a feature space and clustered into groups of similar frames, defining *artificial phonetic categories*, which may or may not coincide with usual phonemes. Acoustical parameters of the source sentence are then mapped within each category, according to similarity of source and target frames.

Another distinctive aspect of VC systems is related to the *phonetic content* of the languages used in training and in actual conversion. In voice conversion within a single language, both text-dependent and text-independent trainings are feasible, and artificial phonetic classes are more likely to reflect actual phonetic classes, since the sets of phonemes present in source and target recordings are basically the same.

On the other hand, *Crosslingual Voice Conversion* [7] assumes that source and target subjects speak different languages (A and B, respectively), and a sentence (in lan-

guage A) from the source speaker should sound as if spoken (untranslated, of course) by the target speaker. This process involves a text-independent training strategy, and is predicated on the assumptions that similar phonemes exist within the languages A and B, and that substitution by similar phonemes would not instantly prohibit comprehension of the converted speech.

Some attempts at crosslingual conversion have been made using bilingual individuals [8], since they allow text-dependent training, by using sentences in language B spoken by the source speaker. This allows the specification of timbral transformations between similar phonemes (in language B) from source and target, which are later applied to phonemes in language A to obtain similar phonemes with the timbre of the target.

1.1 Applications

There has been an increasing interest in voice conversion systems, specially in telecommunication companies such as *CENET* [9]. Some of the applications of voice conversion with a commercial interest are:

- Customization of Text-To-Speech (TTS) interactive systems [6, 10].
- Personalized virtual interpreters: this is a combination of speech recognition followed by automatic translation and finally TTS in the destination language using the voice of the original speaker. Some examples are the **Verbmobil** project (German/English and German/Japanese real-time voice translation), and **TC-STAR** [11, 12, 6, 13, 14, 15, 16, 17] which aims at providing *Speech-to-Speech Translation (SST)* in several languages.
- Biometric voice authentication systems: the development of voice conversion techniques has a natural interplay with the development of voice authentication systems, which are subject to attacks (in this case using voice disguise) as any other security system.
- Voice restoration systems: these are aimed at people who suffered some voice-impairing pathology.

1.2 A Typical Voice Conversion System

Figure 1 presents a sketch of a typical voice conversion system. The system receives sentences from the source speaker (S_0) and from the target speaker (T_0), which are used in a training phase to define a transformation (T) from source speaker features (which may be local or global) to target speaker features. Afterwards, the system receives a new sentence S_f from the source speaker and synthesizes a sentence T_f, which should carry the same message as S_f but with the vocal qualities of the target speaker.

The training phase is generally the first necessary step for voice conversion. This is the stage where data from both speakers is collected and processed, in order to obtain a reasonable characterization of the acoustic features

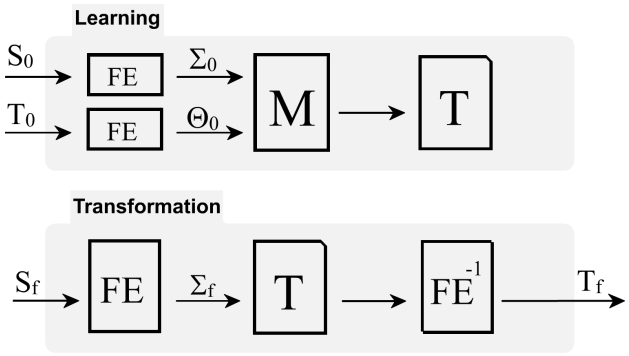


Figure 1. Typical Voice Conversion System.

of each speaker, thus allowing the definition of the transformation to be used in the subsequent stage. According to figure 1, in this phase the VCS:

1. Receives input sentences S_0 and T_0.
2. Extracts relevant acoustic features for each speaker, creating alternative representations Σ_0 and Θ_0 for source and target speakers, respectively.
3. Processes the acoustic features Σ_0 and Θ_0 in order to obtain a database of local (frame-based) and global (sentence-based) descriptors for both speakers.
4. Defines a transformation from local and global source descriptors to local and global target descriptors.

The transformation phase has a similar structure: input sentences S_f are represented as Σ_f in an acoustical feature space, which is then converted by the transformation defined in the training phase into a representation Θ_f, which is finally inverse transformed into a sentence T_f.

The alternative representation of the sentences in a space of acoustical features (Σ and Θ in the diagram) is supposed to preserve enough information so as to allow not only plain resynthesis but also manipulation of timbral and prosodic aspects of the signal. Since many of these are time-varying attributes of the signal, the extraction of acoustic features is usually done on a frame-by-frame basis. Acoustic descriptors are said to be local if they describe a feature of a single frame, and global if they correspond to a whole sentence or to a model of the speaker. Examples of local descriptors are instantaneous pitch, energy, and spectral envelope, or the artificial phonetic category to which a particular frame belongs. Examples of global descriptors are means and standard deviations of pitch or energy measurements, or estimates of the glottal pulse and vocal tract for each speaker.

There is no general consensus with respect to frame size. In theory, frames smaller than $10\ ms$ may be considered stationary due to the inertia of larynx and vocal tract within such timespans [18]. In practice, frames of $15\ ms$ or $25\ ms$ are frequently used [19, 20], and are considered stationary in a broader sense. Some authors [7, 13] prefer to use variable-sized frames defined by an integral number of periods of the quasi-stationary voice signal, which are called *pitch-synchronous frames*.

The choice of frame size is also related to the choice of sample rate, since both combined determine spectral accuracy. In theory, sample rates should never be smaller than 6 kHz, since the human voice has important formant frequencies below 3 kHz, but in practice the sample rates of 8 kHz and 16 kHz are frequently used, and larger sample rates are advised for high-quality voice conversion.

The reconstruction of a voice signal from processed frames must be carefully planned, since changes in spectral content may introduce audible artifacts due to phase differences between adjacent frames, and may be perceived as high-frequency noise, clicks or ringing frequencies that did not exist in the input signal.

The following section presents a historical overview of articles dealing with voice conversion, and also the main techniques introduced. Section 3 presents a comparative overview of these contributions based on perceptual tests. Section 4 discusses some of the frequent difficulties faced when developing VC systems, and some of the proposed solutions. Finally, section 5 summarizes possible future directions in VC development, as presented in the recent literature, and discusses comparison standards for future VC systems.

2. STATE-OF-ART TECHNIQUES IN VOICE CONVERSION SYSTEMS

This section brings a historical overview of techniques used in Voice Conversion, as well as a classification of the techniques according to their interrelations.

2.1 Historical Overview of VC techniques

Many voice conversion (VC) techniques have been proposed since the original formulation of the voice conversion problem by Childers et al. [2] in 1985. Childers proposed solution involved a mapping of acoustical features from a source speaker to a target speaker. A year later, Shikano [21] proposes to use *vector quantization (VQ)* techniques and *codebook* sentences.

A few years later, in 1990, Abe [19] introduces the idea of crosslingual voice conversion systems (CVCS) using bilingual subjects, and in 1991, Valbret [22] rekindles the discussion by proposing personalized *Text-to-Speech* systems using the idea of *Dynamic Frequency Warping (DFW)*.

In 1995, Childers [20] introduces the idea of VC based on the physiological model of glottal pulse and vocal tract, and Narendranath [23] adds *Artificial Neural Networks (ANN)* to the list of VC techniques.

By the end of the 90's, Arslan [24] proposes a model using *Line Spectral Frequencies* for spectral envelope representation, which results in the *STASC (Speaker Transformation Algorithm using Segmental Codebooks)* algorithm, and Stylianou [9] uses *Gaussian Mixture Models (GMMs)* combined with *Mel-Frequency Cepstral Coefficients (MFCCs)* as an alternative to spectral envelope representation.

In 2001, Toda [25] proposes a combined spectral representation and voice conversion technique named *STRAIGHT (Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum)*, which allows the manipulation of spectral, acoustical and rhythmic parameters. A year later Türk [26] proposes a variation of Arslan's *STASC* algorithm using the *Discrete Wavelet Transform (DWT)*.

Sündermann [7] made a series of contributions since 2003. He has established the concept of text-independent voice conversion and has been the first to propose a text-independent crosslingual voice conversion system that did not require bilingual subjects for training the system. He also brought up to the field of voice conversion a technique known as *Vocal Tract Length Normalization (VTLN)*, which had been originally proposed in 1995 by Kamm et al. [27] in the context of speech recognition.

More recent contributions by Rentzos [28], Ye [29] [29], Rao [30] and Zhang [31] have focused in probabilistic techniques, such as *GMMs*, *codebook sentences*, and techniques such as *ANN* and *DFW*, among others.

In the next section, a more detailed view of voice signal representation and transformation techniques frequently used in voice conversion systems is given.

2.2 Classification of VC techniques

Voice conversion techniques may be classified according to the acoustical features used in the alternative representation of the signals, as well as according to the transformation techniques employed in conversion.

2.2.1 Representation Models

There are a few parameters that are usually computed for each frame, such as pitch (F0), energy (rms), and some representation of frequency content, which is fundamental both for classification and transformation of voice quality. Besides the Fourier spectrum and its envelope, voice conversion systems use many other representation models for a voice signal, such as:

Voice-based models: Vocal Tract Length Normalization (VTLN), Formant Frequencies, and Glottal Flow models.

Mixed Voice/Signal Models: Linear Prediction Coding (LPC), Line Spectral Frequencies (LSF), Cepstral Coefficients, and Speech Transformation and Representation using Adaptative Interpolation of weiGHTed spectrum (STRAIGHT).

Signal-based models: Improved Power Spectrum Envelope (IPSE), Discrete Wavelet Transform (DWT), Harmonic plus Noise Model (HNM).

Voice-based models are based on representations of human voice-producing mechanisms, using concepts such as glottal pulse, which is the raw signal produced by vocal folds, and vocal tract, which comprises the oral and nasal cavities, palate, tongue, jaw and lips, and is responsible for many timbral voice qualities.

Mixed voice/signal models are actually signal models that provide compact representations for the signals. Since

they are largely used by the speech recognition community, they acquired many helpful voice-related interpretations. For instance, parts of the cepstrum are often related to formant regions (and thus to vocal tract contribution) or to the fundamental frequency, and LPC coefficients and LPC residuals can also sometimes be associated to vocal tract and glottal pulse (viewed in a subtractive synthesis context).

Purely signal-based models are based on general time-domain and frequency-domain signal representations, and are usually devoid of specific voice-related or phonetic-related semantics. The harmonic-plus-noise model is more specific than the others, but is specially useful in tracking voiced portions (e.g., vowels) of the signal.

Besides the usual linear frequency spacing that is common to Fourier-based methods, many of these techniques also allow the use of alternative frequency scales such as *BARK* and *MEL* [32].

2.2.2 Transformation Techniques

The transformation phase in voice conversion systems is concerned with every acoustic feature used in the representation of the voice signal. This includes pitch shifting and energy compensation, but also the transformation of frequency content in such a way that both timbral aspects and intelligibility are taken into account. Transformation techniques are intrinsically tied to representation models. Some of the common techniques are:

Statistical Techniques: Gaussian Mixture Models (GMM), Hidden Markov Models (HMM), Multi-Space Probability Distributions, Maximum Likelihood Estimators (MLE), Principal Component Analysis (PCA), Unit Selection (US), Frame Selection (FS), K-means, K-histograms.

Cognitive Techniques: Artificial Neural Networks (ANN), Radial Basis Function Neural Networks (RBFNN), Classification And Regression Trees (CART), Topological Feature Mapping, and Generative Topographic Mapping.

Linear Algebra Techniques: Bilinear Models, Singular Value Decomposition (SVD), Weighted Linear Interpolations (WLI) and Perceptually Weighted Linear Transformations, and Linear Regression (LRE, LMR, MLLR).

Signal Processing Techniques: Vector Quantization (VQ) and Codebook Sentences, Speaker Transformation Algorithm using Segmental Codebooks (STASC), and Frequency Warping (FW, DFW, WFW).

These techniques basically differ with respect to the way they look at data. For instance, statistical techniques usually assume that data such as feature vectors or vocal parameters have a random component and may be reasonably described by means and standard deviations (Gaussian model), or that they evolve over time according to simple rules based on the recent past (Markov models).

Cognitive techniques are based on learning processes using abstract neuronal structures, and usually depend on a training phase (where both inputs and outputs are available). Frequently they are used for decision problems (where only 2 possible output values are available), for instance in speech recognition, where a separate network is trained for every specific phoneme or word or sentence that is going to be recognized.

Linear algebra techniques are based on geometrical interpretations of data, for instance in finding simplified models by orthogonal projection (linear regression), in obtaining convex combinations of input data (weighted interpolations), or in decomposing transformations into orthogonal components (SVD).

Signal processing techniques define transformations based on time-domain or frequency-domain representations of the signal. These may try to encode a signal using a library of frequently found signal segments or codewords, or to convert timbre-related voice qualities by modifying frequency scale representations (warping).

The above categories also frequently overlap in the case of voice-conversion techniques. For instance, LRE and SVD are frequently used as statistical tools, and both PCA and vector quantization are built using linear algebra framework.

In the next section some of the usual methods for evaluating voice conversion systems are presented and discussed.

3. EVALUATIONS

The two fundamental questions related to the evaluation of voice conversion addresses the ideas of *similarity* of the timbre of the converted voice with respect to the target voice, and of the *quality* of the result with respect to sound artifacts or intelligibility. Several attempts have been made in trying to answer those questions both objectively and subjectively.

3.1 Objective Evaluation

In this setting it is required that the target sentence be also recorded by the target speaker, thus providing a golden standard to which the converted sentence is compared. Both target and converted sentences are time-aligned and then a global distance between the time-aligned sentences is computed. This global distance can be computed by accumulating differences between time-aligned frames, or using other acoustical measures, such as cepstral distortion (CD) [25], among others.

It has been observed that such objective measures are not necessarily correlated to human perception or to human preferences [33]. In fact, some works report large objective distances and good subjective evaluations [26].

3.2 Subjective Evaluation

Due to the difficulty in defining good objective distance measures which are perceptually meaningful, and also due to the difficulty in comparing objective values using different metrical distances, some authors have preferred to evaluate the performance of their systems by standard subjective tests such as *MOS* and *ABX*.

The *MOS* or *Mean Opinion Score* test is basically an evaluation process using 5 values for grading the output, in this case the quality of the converted voice and its similarity to the target voice. The values are standardized as **5**=excellent, **4**=good, **3**=fair, **2**=poor, **1**=bad. The project TC-STAR [11, 12] proposes a standard perceptual test using *MOS* as a measure of both quality and similarity.

Tables 1 and 2 bring a collection of MOS results for quality and similarity, respectively, of several voice conversion systems, as presented by their authors. In experimental voice conversion tests, a distinction is usually made between *intra-gender* conversion (indicated by $M \rightarrow M$ and $F \rightarrow F$ in the tables) and *inter-gender* conversion ($M \rightarrow F$ and $F \rightarrow M$). The method of Rao [30] has received the highest grading for quality of conversion, whereas Rentzos [28] methods have been graded higher for similarity of timbres. Some authors [16, 3] reported that variants of their methods were able to significantly improve quality gradings at the expense of lower similarity gradings, probably due to excessive smoothing issues (see section 4).

Year	Author	Quality *MOS*	Technique
1997	Kim [34]	3.42	VQ
1998	Kain [10]	4.20 ($M \rightarrow M$) 2.70 ($M \rightarrow F$)	GMM
1998	Stylianou [9]	≈ 2.70	GMM
2001	Toda [25]	≈ 4.20 ($M \rightarrow M$) ≈ 2.70 ($F \rightarrow F$)	GMM DFW
2004	Pfitzinger [5]	≈ 1.50	WLI
2005	Toda [35]	≈ 3.10 ($F \rightarrow M$) ≈ 3.30 ($M \rightarrow F$)	MLE
2006	Nurminen [14]	2.09	GMM
2006	Duxans [6]	2.37	GMM CART
2006	Sündermann [17]	2.7 (Txt-Dependent) 2.6 (Txt-Independent)	US
2006	Rao [30]	4.56 ($M \rightarrow F$) 4.71 ($F \rightarrow M$)	WLI
2006	Shuang [15]	4.09 (UK English) 3.68 (CN Mandarin)	FW
2007	Dutoit [3]	2.56	FS
2007	Erro [13]	3.27 ($M \rightarrow M$) 3.00 ($M \rightarrow F$) 3.60 ($F \rightarrow M$) 4.20 ($F \rightarrow F$)	WFW
2007	Fujii [4]	3.03 ($F \rightarrow F$) 2.75 ($M \rightarrow F$)	US
2008	Shuang [16]	3.48	FW
2008	Zhang [36]	3.00 ($M \rightarrow M$) 2.70 ($M \rightarrow F$) 3.10 ($F \rightarrow M$) 2.80 ($F \rightarrow F$)	VQ
2008	Desai [37]	≈ 2.70	ANN
2009	Zhang [31]	2.70 ($F \rightarrow M$) 2.50 ($F \rightarrow F$)	VQ

Table 1. Experimental Results for Quality *MOS* in Voice Conversion systems.

These tests have all been made using source and target speakers of the same language. Some MOS results for crosslingual conversion between English and Spanish have been reported by Duxans [6] in 2006. In his study, *MOS* gradings for quality were 2.37 for Spanish-to-Spanish conversion and 2.33 for Spanish-to-English conversion, and similarity MOS gradings were 3.18 (Spanish-to-Spanish) and 2.79 (Spanish-to-English).

Year	Author	Similarity *MOS*	Technique
2003	Rentzos [28]	3.65	HMM
2006	Nurminen [14]	3.10 ($F \rightarrow F$) 3.05 ($F \rightarrow M$) 2.20 ($M \rightarrow F$) 1.77 ($M \rightarrow M$)	GMM
2006	Duxans [6]	3.18	GMM CART
2006	Rao [30]	2.92 ($M \rightarrow F$) 3.23 ($F \rightarrow M$)	WLI
2006	Shuang [15]	1.87 (UK English) 2.77 (CN Mandarin)	FW
2007	Dutoit [3]	2.77	FS
2007	Erro [13]	2.93 ($M \rightarrow M$) 3.27 ($M \rightarrow F$) 2.53 ($F \rightarrow M$) 3.00 ($F \rightarrow F$)	WFW
2008	Shuang [16]	2.20	FW
2008	Zhang [36]	2.20 ($M \rightarrow M$) 2.30 ($M \rightarrow F$) 2.50 ($F \rightarrow M$) 2.10 ($F \rightarrow F$)	VQ

Table 2. Experimental Results for Similarity *MOS* in Voice Conversion systems.

The *ABX* test is a two-alternative test that is often used in comparing similarity between converted and target sentences. In this test, experimental subjects have to decide whether a given sentence X is closer in vocal quality to one of a pair of sentences A and B, where one of them is the source and the other is the target, not necessarily in that order. Success is measured by the percentage of answers of the type $X \approx T$ where $T \in \{A, B\}$ is the target.

Table 3 shows a set of *ABX* results of several voice conversion systems, as reported by their authors. Among these, a method by Fujii [4] stands out with extremely high scores. The main problem with interpreting *ABX* scores is the fact that subjects are not allowed to answer that the sentence X is not similar to neither A nor B, if that is the case. It can be inferred that a method that is successful according to an *ABX* test might in fact have a very low similarity-*MOS* value, as long as the similarity of the X sentences to their respective source speakers were even lower.

Another use of the *ABX* test is the comparison of two different techniques applied to the same problem. In this setting X is the target sentence, and A and B are the converted sentences using both techniques. Subjects are required to answer which of the sentences is closer to the target, but here the subject is allowed to answer "neither". Success rates are computed for each technique as the percentage of sentences for which that technique has been chosen as closer to the target.

Among authors using this type of test are Pozo [42] and Desai [37]. Pozo compared his *Joint Estimation Analysis-Synthesis (JEAS)* method to the *Pitch-Synchronous Harmonic Model (PSHM)*, obtaining the following success rates: 41% ($M \rightarrow M$), 37% ($M \rightarrow F$), 33% ($F \rightarrow M$) and 36.5% ($F \rightarrow F$). Desai [37] compared his method using ANN to traditional GMM methods, obtaining an ABX

Year	Author	ABX Index	Technique
1998	Kain [10]	52.5% $(M \rightarrow M)$ 97.5% $(M \rightarrow F)$	GMM
1998	Stylianou [9]	97%	GMM
1999	Arslan [24]	78% $(M \rightarrow M)$ 100% $(M \rightarrow F)$	STASC
2001	Toda [25]	$\approx 77\%$ $(M \rightarrow M)$ $\approx 83\%$ $(F \rightarrow F)$	GMM DFW
2004	Orphanidou [38]	79.5% $(M \rightarrow M)$ 86.3% $(M \rightarrow F)$ 88.6% $(F \rightarrow M)$ 77.3% $(F \rightarrow F)$	RBFNN
2005	Toda [35]	$\approx 84\%$ $(M \leftrightarrow F)$	MLE
2005	Zhang [39]	87.5%	GMM
2006	Ye e Young [29]	91.8%	GMM
2007	Fujii [4]	100% $(M \rightarrow M)$ 100% $(M \rightarrow F)$ 100% $(F \rightarrow M)$ 98.0% $(F \rightarrow F)$	US
2007	Hanzlicek [40]	87.2% $(F \rightarrow M)$ 70.8% $(F \rightarrow F)$	GMM
2008	Yue [41]	92.0%	GMM HMM
2008	Zhang [36]	$\approx 62\%$ $(M \rightarrow M)$ $\approx 80.5\%$ $(M \rightarrow F)$ $\approx 78.5\%$ $(F \rightarrow M)$ $\approx 55\%$ $(F \rightarrow F)$	VQ
2009	Zhang [31]	68% $(M \rightarrow F)$ 84% $(F \rightarrow F)$	VQ

Table 3. Experimental Results for *ABX* indices in Voice Conversion systems.

success rate for similarity of 65.0%. Likewise, Türk [26] compared his *Subband* conversion to *Full-band* methods, reporting an *ABX* index of 92.9% for similarity.

Comparing empirical results such as those reported in this section without considering the details of the experimental settings makes little (if any) sense. There are many concurrent factors that can significantly influence the outcome of an experiment, such as the number of sentences, number of subjects, subject listening sensitivity, original audio quality, unambiguity of the questions, among many others.

For instance, experimental settings must be carefully defined and questions must be carefully explained in order to obtain consistent experimental data. The description of the experiment should be detailed enough so as to enable independent replication of experimental results. Also, statistical analysis should be taken seriously, including hypothesis testing and measured significance levels, in order to derive statistically significant conclusions.

Another related difficulty is the lack of a standard database for voice conversion. For instance, Zhang [39] uses the *MSRA Mandarin Database*, Toda [35] uses the *MOCHA Database*, Ye e Young [29] uses the *VOICES Database*, whereas Türk [26] and Guido [43] use the *TIMIT Database*, which is probably the most popular choice.

These observations reinforce the need for some kind of *Benchmark* for subjective experimental evaluations of voice conversion systems, enabling researchers to set up similar experiments and to obtain comparable experimental data. This would benefit the community by helping to correctly identify advantages and limitations of each technique, which are the subject of the next section.

4. LIMITATIONS AND CHALLENGES

There are many open problems in voice conversion, which have been identified in several previous articles:

Phonetic Issues: in crosslingual voice conversion it is well-known that many phonemes in the source language may not exist in the target language. Bilingual subjects have been used [8] in order to derive phonetic transformations that allow similar but not identical phonemes to be converted between languages. Whether such transformations might be successfully used in other (not bilingual) subjects is an open question.

Prosody Issues: there are many global acoustic aspects that are decisive in order to obtain good conversions, such as average pitch and standard pitch deviation, average and standard deviation of energy, statistics related to the rhythmic flow of speech, and so on. Some (but not all) of these issues are discussed in Helander [44] and Hanzlicek [45].

Quality Issues: some of the problems that are perceived as a lack of quality are hissing noise, ringing tones, clicks and also timbral aspects that may be described as a synthetic or unnatural voice. For instance, large pitch shifts without formant correction may degrade quality (and even intelligibility) of the converted voice. These issues have been reported many times [9, 10, 4] and are easily detected in subjective tests.

Similarity Issues: these are related to the timbral quality and vocal identity, mainly correlated to phonetic aspects of speech, although they may easily be confused with quality and prosody issues in experimental tests. In theory, a purely synthetic voice might be perceived as unnatural but similar in timbre to a target voice. Intergender voice conversion is particularly susceptible to this type of problem [10, 23]

Evaluation Issues: this has been discussed in the previous section. Objective measures such as spectral distance or cepstral distortion may be uncorrelated to human perceptual measures [33, 26], whereas subjective measures such as *MOS* or *ABX* may be useful if some sort of experimental benchmark is agreed upon.

Excessive Smoothing Issues: this is a technical issue caused by interpolation methods in the transformation phase, which degrade the spectrum by eliminating details and reducing the similarity of target and converted voices [35, 46, 47].

Overfitting Issues: this is a counterpart of the previous issue, and is caused by using excessive data in training and obtaining an excessively fine-grained transformation which might produce discontinuities between adjacent frames [47].

5. CONCLUSION

This text has presented the voice conversion problem and discussed some of its application contexts, such as TTS customization [10, 6] and virtual interpreters [11, 12]. Major contributions in the recent literature, as well as comparative results, have also been presented and discussed.

High-level representation models for voice signals are a critical aspect of any voice conversion system, since they define and also constrain the available transformation techniques. Aspects such as timbre, prosody and intelligibility should all be taken into account for better results in terms of naturalness and fluency of virtual interpreters and of customized TTS systems [10, 6]. The challenge of crosslingual voice conversion has brought interest to studies of phonetic similarities and differences across languages and automatic phonetic transformation, specially resorting to bilingual individuals [19, 8].

The choice of training model for a voice conversion system usually depend on specific requirements (for instance, attempts at breaking a voice security system may require text-dependent training in order to minimize conversion errors) or on availability of data (for instance, if crosslingual conversion between non-bilingual subjects is desired, then text-independent training is the only available option).

Transformation techniques should be considered not only in relation to compatible representation models, but also with respect to the prosodic and timbral aspects that will be converted, since they define how a voice conversion system views and manipulates such high-level representation data.

Among the open research problems, the definition of a benchmark for the subjective comparison of voice conversion systems for quality and similarity assessment seems to be one the most urgent issues. Some progress has already been made in this respect through the TC-STAR project [11, 12], but a more thorough specification of reproducible experiments is desirable.

6. REFERENCES

[1] P. R. Cook, "Singing voice synthesis: History, current work, and future directions," *Computer Music Journal*, vol. 20, pp. 38–46, 1996.

[2] D. G. Childers, B. Yegnanarayana, and K. Wu, "Voice conversion: Factors responsible for quality," *ICASSP*, pp. 748–751, 1985.

[3] T. Dutoit, A. Holzapfel, M. Jottrand, A. Moinet, J. Perez, and Y. Stylianou, "Towards a voice conversion system based on frame selection," in *ICASSP*, pp. 513–516, 2007.

[4] K. Fujii, J. Okawa, and K. Suigetsu, "High-individuality voice conversion based on concatenative speech synthesis," *IJCSE*, vol. 2, p. 1, 2007.

[5] H. Pfitzinger, "Unsupervised speech morphing between utterances of any speakers," in *SST 2004*, pp. 545–550, 2004.

[6] H. Duxans, D. Erro, J. Pérez, F. Diego, A. Bonafonte, and A. Moreno, "Voice conversion of non-aligned data using unit selection," in *TC-STAR WSST*, 2006.

[7] D. Sündermann, H. Ney, and H. Hoge, "VTLN-based cross-language voice conversion," in *ASRU*, 2003.

[8] M. Mashimo, T. Toda, H. Kawanami, H. Kashioka, K. Shikano, and N. Campbell, "Evaluation of cross-language voice conversion using bilingual e non-bilingual databases," *Interspeech*, pp. 293–296, 2002.

[9] Y. Stylianou, "Continuous probabilistic transform for voice conversion," *IEEE TSAP*, no. 6, pp. 131–142, 1998.

[10] A. Kain and M. Macon, "Text-to-speech voice adaptation from sparse training data," in *5th ICSLP*, 1998.

[11] D. Sündermann, H. Höge, A. Bonafonte, H. Ney, and J. Hirschberg, "TC-STAR: Cross-language voice conversion revisited," *TC-STAR WSST*, pp. 231–236, 2006.

[12] A. Bonafonte, H. Höge, I. Kiss, A. Moreno, U. Ziegenhain, H. van den Heuvel, H. Hain, X. Wang, and M. Garcia, "TC-STAR: Specifications of language resources and evaluation for speech synthesis," in *LREC*, 2006.

[13] D. Erro and A. Moreno, "Weighted frequency warping for voice conversion," in *Interspeech*, 2007.

[14] J. Nurminen, V. Popa, J. Tian, Y. Tang, and I. Kiss, "A parametric approach for voice conversion," in *TC-STAR WSST*, pp. 225–229, 2006.

[15] Z. Shuang, R. Bakis, and Y. Qin, "Voice conversion based on mapping formants," in *TC-STAR WSST*, pp. 219–223, 2006.

[16] Z. Shuang, R. Bakis, and Y. Qin, "IBM voice conversion systems for 2007 TC-STAR evaluation," *Tsinghua Science & Technology*, vol. 13, no. 4, pp. 510–514, 2008.

[17] D. Sündermann, H. Hoge, A. Bonafonte, H. Ney, A. Black, and S. Narayanan, "Text-independent voice conversion based on unit selection," in *ICASSP*, 2006.

[18] H. Kawahara and T. Irino, "Exploring temporal feature representations of speech using neural networks," Technical Report SP88-31 (in Japanese), IEICE, Tokyo, 1988.

[19] M. Abe, K. Shikano, and H. Kuwabara, "Cross-language voice conversion," in *ICASSP*, pp. 345–348, 1990.

[20] D. Childers, "Glottal source modeling for voice conversion," *Speech Communication*, vol. 16, pp. 127–138, 1995.

[21] K. Shikano, K. Lee, and R. Reddy, "Speaker adaptation through vector quantization," in *ICASSP*, vol. 11, 1986.

[22] H. Valbret, E. Moulines, and J. Tubach, "Voice tranformation using PSOLA technique," in *2nd ECSCT*, 1991.

[23] M. Narendranath, H. Murthy, S. Rajendran, and B. Yegnanarayana, "Transformation of formants for voice conversion using artificial neural networks," *Speech Communication*, vol. 16, pp. 207–216, 1995.

[24] L. Arslan, "Speaker transformation algorithm using segmental codebooks (STASC)," *Speech Communication*, vol. 28, no. 3, pp. 211–226, 1999.

[25] T. Toda, H. Saruwatari, and K. Shikano, "Voice conversion algorithm based on gaussian mixture model with dynamic frequency warping of straight spectrum," *Power [dB]*, vol. 30, p. 40, 2001.

[26] O. Türk and L. Arslan, "Subband based voice conversion," in *7th ICSLP*, 2002.

[27] T. Kamm, G. Andreou, and J. Cohen, "Vocal tract normalization in speech recognition: Compensating for systematic speaker variability," in *15th ASRS*, 1995.

[28] D. Rentzos, S. Vaseghi, E. Turajlic, Q. Yan, and C. Ho, "Transformation of speaker characteristics for voice conversion," in *IEEE WASRU*, pp. 706–711, 2003.

[29] H. Ye and S. Young, "Quality-enhanced voice morphing using maximum likelihood transformations," *IEEE TASLP*, vol. 14, no. 4, pp. 1301–1312, 2006.

[30] K. Rao and B. Yegnanarayana, "Voice conversion by prosody and vocal tract modification," in *9th ICIT*, pp. 111–116, 2006.

[31] M. Zhang, J. Tao, J. Nurminen, J. Tian, and X. Wang, "Phoneme cluster based state mapping for text-independent voice conversion," in *ICASSP*, pp. 4281–4284, 2009.

[32] F. Nolan, "Intonational equivalence: an experimental evaluation of pitch scales," in *ICPhS*, pp. 771–774, 2003.

[33] D. Sündermann, "Voice conversion: State-of-the-art and future work," *FORTSCHRITTE DER AKUSTIK*, vol. 31, p. 735, 2005.

[34] E. Kim, S. Lee, and Y. Oh, "Hidden markov model based voice conversion using dynamic characteristics of speaker," in *5th ECSCT*, 1997.

[35] T. Toda, A. Black, and K. Tokuda, "Spectral conversion based on maximum likelihood estimation considering global variance of converted parameter," in *ICASSP*, pp. 9–12, 2005.

[36] M. Zhang, J. Tao, J. Tian, and X. Wang, "Text-independent voice conversion based on state mapped codebook," in *ICASSP*, 2008.

[37] S. Desai, E. Raghavendra, B. Yegnanarayana, A. Black, and K. Prahallad, "Voice conversion using artificial neural networks," in *IEEE WSLT*, 2008.

[38] C. Orphanidou, I. Moroz, and S. Roberts, "Wavelet-based voice morphing," *WSEAS Transactions on Systems*, vol. 3, no. 10, pp. 3297–3302, 2004.

[39] J. Zhang, J. Sun, and B. Dai, "Voice conversion based on weighted least squares estimation criterion and residual prediction from pitch contour," *Lecture notes in computer science*, vol. 3784, p. 326, 2005.

[40] Z. Hanzlíček and J. Matoušek, "Voice conversion based on probabilistic parameter transformation and extended inter-speaker residual prediction," *Lecture Notes in Artificial Intelligence*, vol. 4629, pp. 480–487, 2007.

[41] Z. Yue, X. Zou, Y. Jia, and H. Wang, "Voice conversion using HMM combined with GMM," in *CISP'08*, vol. 5, 2008.

[42] A. del Pozo and S. Young, "The linear transformation of LF glottal waveforms for voice conversion," in *Interspeech*, pp. 1457–1460, 2008.

[43] R. Guido, L. Sasso Vieira, S. Barbon Júnior, F. Sanchez, C. Dias Maciel, E. Silva Fonseca, and J. Carlos Pereira, "A neural-wavelet architecture for voice conversion," *Neurocomputing*, vol. 71, no. 1-3, pp. 174–180, 2007.

[44] E. Helander and J. Nurminen, "A novel method for prosody prediction in voice conversion," in *ICASSP*, vol. 4, pp. 509–512, 2007.

[45] Z. Hanzlicek and J. Matousek, "F0 transformation within the voice conversion framework," in *ICSLP*, pp. 1961–1964, 2007.

[46] T. Toda, A. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE TASLP*, vol. 15, no. 8, p. 2222, 2007.

[47] L. Mesbahi, V. Barreaud, and O. Boeffard, "Comparing GMM-based speech transformation systems," in *Interspeech*, pp. 1989–1992, 2007.

[48] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, "Voice conversion through vector quantization," in *ICASSP*, pp. 655–658, 1988.

[49] O. Türk and M. Schröder, "A comparison of voice conversion methods for transforming voice quality in emotional speech synthesis," *Interspeech*, 2008.

[50] H. de Paula, H. Yehia, D. Shiller, G. Jozan, K. Munhall, and E. Vatikiotis-Bateson, "Linking production and perception through spatial and temporal filtering of visible speech information," in *6th ISSP*, pp. 37–42, 2003.

[51] L. Arslan and D. Talkin, "Voice conversion by codebook mapping of line spectral frequencies and excitation spectrum," in *5th ECSCT*, 1997.

[52] Y. Stylianou, "Voice transformation: A survey," *ICASSP*, pp. 3585–3588, 2009.

AUTOMATIC SONG COMPOSITION FROM THE LYRICS EXPLOITING PROSODY OF JAPANESE LANGUAGE

Satoru Fukayama, Kei Nakatsuma, Shinji Sako, Takuya Nishimoto and Shigeki Sagayama
the University of Tokyo, Nagoya Institute of Technology,
{fukayama,nishi,sagayama}@hil.t.u-tokyo.ac.jp
tsuma@alab.t.u-tokyo.ac.jp, s.sako@nitech.ac.jp

ABSTRACT

Automatic composition techniques are important in sense of upgrading musical applications for amateur musicians such as composition support systems. In this paper, we present an algorithm that can automatically generate songs from Japanese lyrics. The algorithm is designed by considering composition as an optimal-solution search problem under constraints given by the prosody of the lyrics. To verify the algorithm, we launched *Orpheus* which composes with the visitor's lyrics on the web-site, and 56,000 songs were produced within a year. Evaluation results on the generated songs are also reported, indicating that *Orpheus* can help users to compose their own original Japanese songs.

1. INTRODUCTION

Recently, there has been wide interest in automatic composition algorithms which can help amateur musicians to compose original tunes. Although considerable amount of research has been done on automatic composition[1][2][3], much less has been done on composing songs from the lyrics, and the question of what information in the lyrics should be exploited for generating songs remains. Syntactic information of the lyrics are used in some researches[4] to compose a song from the lyrics. Musicologists argue that there are considerable correlations between music and prosody. In case of composing songs, prosody plays a more important role. However, no system that uses prosody has yet been attempted.

2. MELODY COMPOSITION ALGORITHM EXPLOITING PROSODY OF JAPANESE LYRICS

Our objective is to develop a method to generate a melody automatically which satisfies the constraints given by the prosody of Japanese lyrics. We define melody composition here as generating a melody given the lyrics, patterns of rhythms (by "rhythm tree" which is described later in this section), and harmony sequence with specifications of tonality and scale.

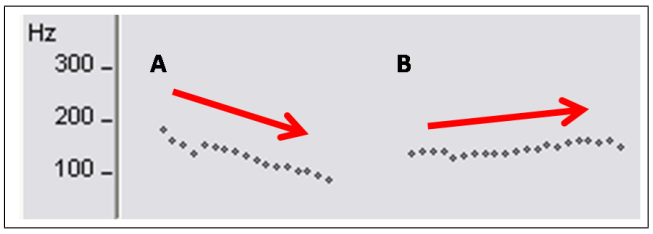


Figure 1. Pitch contuor of "ka'mi" and "kami"': the pitch accent of the Japanese word is lexically contrastive in ka'mi(A) 'god' vs. kami'(B) 'paper'.

The composition algorithm consists of two parts. The former part is for designing the rhythm for the melody by considering the uniformity of the rhythm in the song. The latter is for designing the pitch of each note with probabilistic inference.

We firstly review the properties of Japanese prosody and argue the importance of considering prosodic information when composing Japanese songs. Secondly, we describe a method to generate rhythm of the melody. Then, we discuss how to obtain a proper pitch given the rhythm, harmony sequence, and the lyrics. Finally, we introduce our automatic song composition system *Orpheus*, which is based on our proposed algorithm and used in the evaluations.

2.1 Japanese Prosody and its Role in Composition

Japanese is said to "have a fixed shape consisting of a sharp decline around the accented syllable, a decline that is usually analysed as a drop from a H [1] tone to a L [2] "[5]. Furthermore, "the place of the accent is lexically contrastive, as in ka'mi 'god' vs. kami' 'paper"'[5]. (Fig. 1)

A melody attached to the lyrics cause an effect similar to the accent. Therefore we can assume that the prosody of Japanese lyrics imposes constraits on pitch motions of the melody.

2.2 Composition of Rhythm

In order to design a rhythm on the given lyrics, two problems have to be considered. The first problem is the allocation of lyrics on the melody, and the other problem is how to handle the unity of rhythm in the same song.

[1] H: high
[2] L: low

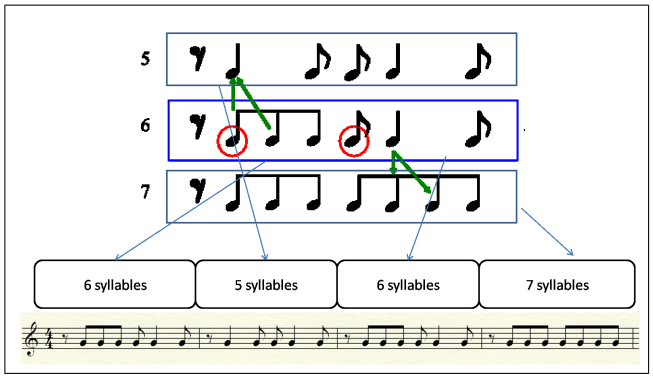


Figure 2. By using the "rhythm tree"(above), rhythm corresponding to the number of syllables are generated with consideration of keeping the unity of rhythm feature in the same song.

2.2.1 Allocation of Lyrics on Melody

In order to solve the first problem, we assumed that melody consists of segments and the lyrics should be divided into these segments. For instance, 2 bars can be handled as a segment for a song with a length of 8 bars. This is because the lyrics input by the user are not always "formatted" as poem like the usual lyrics are.

Furthermore, in most classical Japanese songs, one syllable (mora) corresponds to one note in a melody. Thus the number of notes in each segment is determined by allocating the syllables. When we consider the constraits on dividing the lyrics, the following 3 criteria can be assumed:

- A simillar number of syllables in each segments is prefered.
- The border of the segments should not be crossed over within a word.
- Too short lyrics should be iterated prior to allocation.

Under these constraits, we can solve the syllable allocation problem by using dynamic programming.

2.2.2 Keeping Unity of Rhythm in Melody

Even though the numbers of notes in each of the segments are decided, there still exist a large degree of freedom in rhythm. One possible way to put constraits on rhythm is to arrange that the generated rhythm belogs to a same "family" of rhythm. It is reasonable to assume that the "rhythm family" does not change in a relatively short song.

Here the second problem, that is, the requirement to keep the unity of rhythm in melody arises. To cope with this problem, we introduce a "rhythm tree" that is one rhythm has a simillar feature when one can be derived by uniting or dividing the note just one on the another. In practice, a tree structured templates of rhythm as shown in Fig. 2 are prepared by hand beforhand. Since the number of syllables in each segment corresponds to the number of notes, this tree structured template determine the rhythm in each segment.(Fig. 2)

2.3 Composition of Pitches

2.3.1 Composition with Probabilistic Inference

In this section, we discuss on how to obtain a pitch sequence. Although there are some discussions on the definition of melody, still it is likely to say there are certain tendency in melody. For example, in case of song, pitches of the melody would be constrained by the usual voice range of the singer. The prosody of the lyrics also impose constraints on pitch motions of the melody. As we reviewed at section 2.1, pitch motions of Japanese songs largely follow the up-ward and down-ward motions based on the prosody of the lyrics. Furthermore, chord progression, bass line of the accompaniment part and durations of each notes impose constraints on occurence and transition of pitches on the basis of écuriture of composition, such as harmony and counterpoint. Althogh exploiting these écuritures are not always indispensable to discuss how can we generate a cutting edge contemporary music automatically, still we can assume that these écuritures would secure the quality of generated songs with our algorithm for the purpose of composition support system for amateur musicians.

If a certain melody were obtained, the melody would satisfy these constraits as we disucussed above. Conversely, we can compose a song by finding the melody which optimally meets the condition. Let the pitch sequence as a sequence of MIDI note number be $X_1^N = x_1 x_2 \cdots x_N$, and the sequence of conditions on pitch sequence be $Y_1^N = y_1 y_2 \cdots y_N$, where each y_n involves chord label with annotations of scale and tonality(c_n), duration of the note (d_n), MIDI note number of the accompaniment bass (b_n), and pitch accent information, i.e. $y_n = (c_n, d_n, b_n, a_n)$. Let us also denote $P\left(X_1^N | Y_1^N\right)$ as conditional probability for X_1^N given Y_1^N which represent the tendency of pitch sequences X_1^N under condition Y_1^N. The composition of pitch for melody can be considered as finding an optimal sequence X_1^{N*} given Y_1^N which maximize $P\left(X_1^N | Y_1^N\right)$:

$$X_1^{N*} = \underset{X_1^N}{\operatorname{argmax}} P\left(X_1^N | C_1^N\right). \tag{1}$$

By assuming

$$P\left(x_n | X_1^{n-1}, Y_1^N\right) \simeq P\left(x_n | x_{n-1}, Y_1^N\right), \tag{2}$$

equation (1) will be as follows:

$$X_1^{N*} = \underset{X_1^N}{\operatorname{argmax}} \prod_{n=1}^{N} P\left(x_n | x_{n-1}, Y_1^N\right), \tag{3}$$

where $P\left(x_1 | x_0, Y_1^N\right) = P\left(x_1 | Y_1^N\right)$.

Since there are 128^N possible sequence of pitches, it is computationally unfeasible to search the optimal sequence by calculating probabilities for all of the possible sequences. However, we can obtain the optimal pitch sequence in order $O(N)$ using dynamic programming[6].

2.4 Implementation of the Composition System

Orpheus is an automatic composition system that we implemented using proposed algorithm for melody composition. This system computes melody from the lyrics input with choices of chord progressions, rhythm pattern,

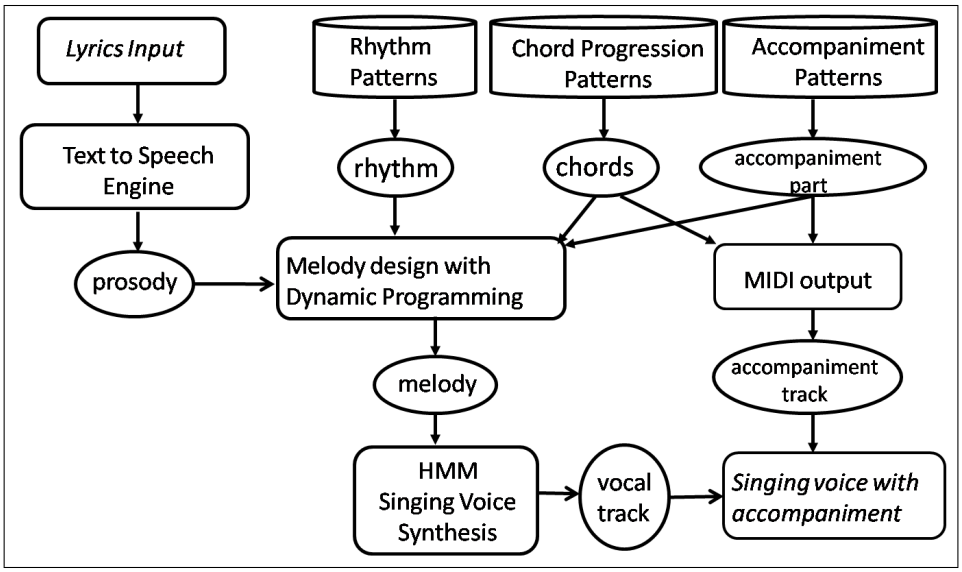


Figure 3. Flow chart of processes: *Orpheus* generates songs with the lyrics input and the choices of patterns.

and accompaniment instruments. Flow chart of the processes is shown in Fig. 3. We used Galatea-Talk[7] text-to-speech engine to analyze the prosody of Japanese lyrics, and HMM singing voice synthesizer[8] to generate the vocal part. We also implemented the system as a web-based application [3] .

3. EVALUATION RESULTS

We did two experiments to evaluate the algorithm. Firstly, we asked a classical music composer to evaluate generated songs in five-grade evaluation. The results on 59 generated songs are shown in Fig. 5 These results indicate that 83.1% of the generated pieces satisfactorily follow classical music theory, and 91.6% of the songs were voted as attractive aside from musical theory. Example of generated song is shown in Fig. 4.

Secondly, we uploaded our system to get comments from a large number of users on the internet. During a year of operation, about 56,000 songs were generated by the users and 1378 people answered the questions about *Orpheus* and the generated songs. Summarization of answers in five-grade evaluation is shown in Fig. 6 Judging from the results, about 70.8% commented that the generated songs are attractive, and 84.9% of the users had fun trying this system.

4. DISCUSSIONS

The evaluations results by a composer indicate that most of the generated songs are able to be called "melody" at least in theory. Songs that are evaluated "very poor" had irregular usages of non-chord tones which rule cannot be described with the relationship between the current note and the previous note.

Evaluation by the users on the internet suggest that our composition system is an enjoyable solution for amateur

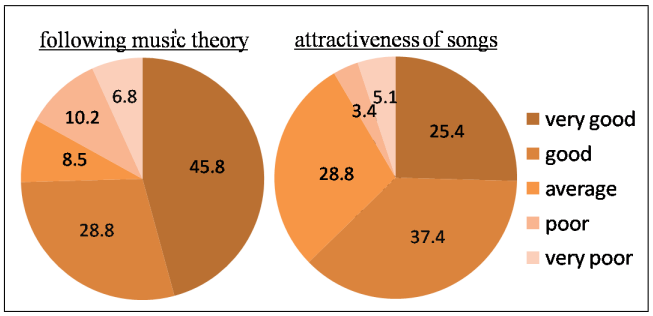


Figure 5. Evaluation results on 59 generated songs by a classical music composer. 83.1% of the generated pieces satisfactorily follow classical music theory, and 91.6% of the songs were voted as attractive aside from musical theory.

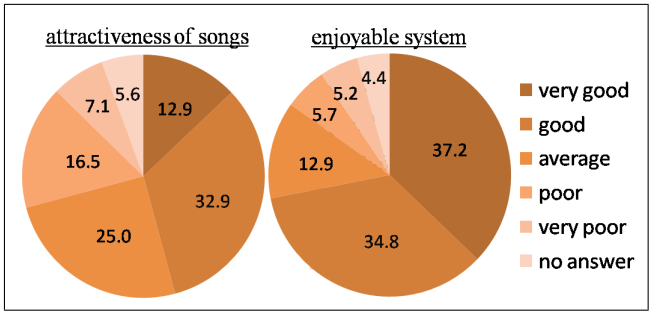


Figure 6. Evaluation results on generated songs and the *Orpheus* by 1378 users. 70.8% commented that the generated songs are attractive, and 84.9% of the users had fun trying this system.

[3] http://orpheus.hil.t.u-tokyo.ac.jp

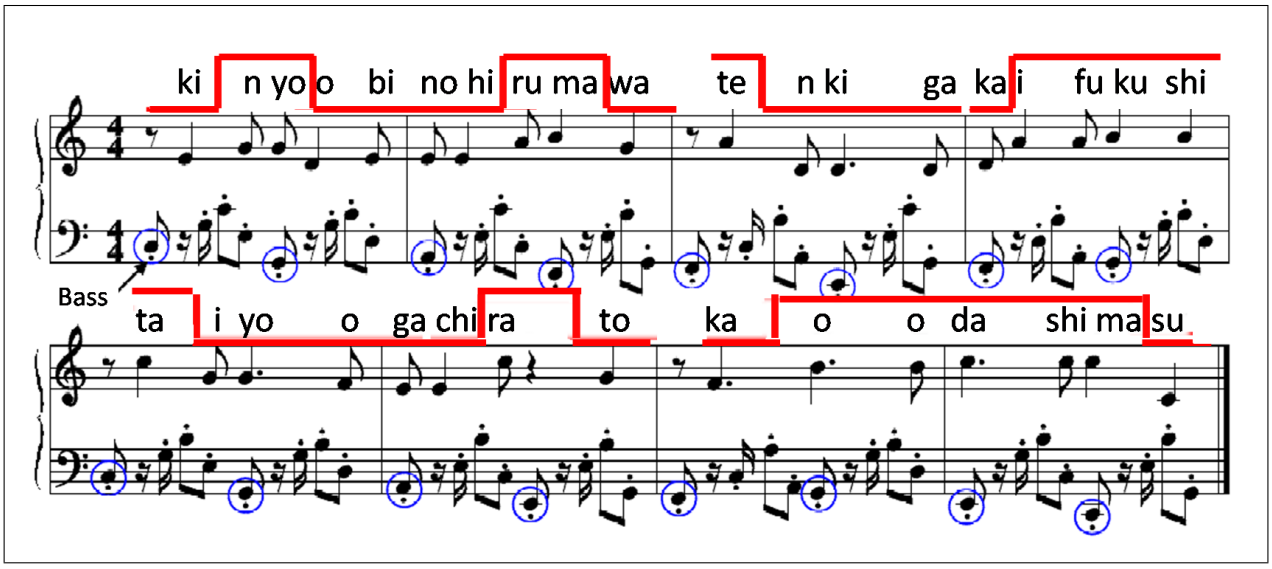


Figure 4. Example of generated song: this song was generated with the lyrics input of weather forecast. The red lines indicates the pitch accents of the Japanese lyrics. The pitch motions of the melody follows the pitch accent of the lyrics.

to compose their original songs. One reason for the result could be the directability on generating songs. Users can generate various melody by typing arbitrary lyrics since the generated song will vary based on the prosody of the lyrics. This may enabled the user not only generating a song automatically but also to generate their original songs with their original lyrics.

5. CONCLUSION

This research attempted to design an algorithm to compose a song automatically from the lyrics using prosody information, which enables users to make their original songs easily. The results indicate that our method and implemented system *Orpheus* is an enjoyable solution for amateur musicians.

However, it should be noted that our algorithm can be applied to lyrics written in "pitch accent" languages only. As a next step, we plan to extend the composition algorithm to handle "stress accent" languages, such as English, by putting constraints on metric structure of the melody.

6. REFERENCES

[1] L. Hiller and L. Isaacson, *Experimental Music*. McGraw-Hill, 1959.

[2] I. Xenakis, *Formalized Music. Revised edition*. Pendragon Press, 1992.

[3] D. Cope, *Computers and Musical Style*. A-R Editions, 1991.

[4] K. H. et al., "Automatic song composer from phrase structure of lyrics," in *Proc. of 57th Information Processing Society of Japan Annual Convention*, pp. 11–12, 1998.

[5] M. E. Beckman and J. B. Pierrehumbert, "Intonational structure in japanese and english," in *Phonology Yearbook 3*, pp. 255–309, 1986.

[6] R. E. Bellman, *Dynamic Programing*. Princeton University Press, 1957.

[7] S. K. et al., "Galatea: Open-source software for developing anthropomorphic spoken dialog agents," in *Life-Like Characters*, pp. 187–212, Springer-Verlag, 2004.

[8] S. S. et al., "A singing voice synthesis system based on hidden marcov model," in *Transactions of Information Processing Society of Japan*, pp. 719–727, 2004.

MIMICRY OF TONE PRODUCTION: RESULTS FROM A PILOT EXPERIMENT

Tommaso Bianco
IRCAM, CNRS - UMR STMS, Paris
tommaso.bianco@ircam.fr

Marcelo M. Wanderley
Idmil, McGill University, Montreal
marcelo.wanderley@mcgill.ca

Frederic Bevilacqua
IRCAM, CNRS - UMR STMS, Paris
frederic.bevilacqua@ircam.fr

ABSTRACT

In this paper we present the description and the first results of a pilot experiment in which participants were requested to mimic the production of sonic elements trough different control modalities. Results show different degrees of dependence of the control temporal profiles with the dynamic level and temporal ordering of the stimuli. The protocol and methodology here advanced may turn useful for ameliorating existing mapping strategies for gesture based interactive media, with particular emphasis to adaptive control of physics-based models for sound synthesis.

1. INTRODUCTION

The decoupling between control input and acoustical output in computer music instruments opened a problematic that still remains central for research in the musical domain: the combination between ease of use and effectiveness, two fundamental components of system's usability [1]. If we consider the case of control for virtual acoustical instruments based on physical modeling, this combination can become even more baffling. The user may indeed be compelled to control an instrument without the interaction feedback, and with a physical interface that bases on a control modality far different from the real case.

The connection between user's intention and sonic outcome has been previously tackled at different levels: at a physical and physiological level for the choice of the machine transducers [2] [3], at gestural level for the definition of tasks and evaluation of performance [4] or evaluation of similarities [5], and at a cognitive level, for the understanding of mental coding of musical experience though motor-mimetic imagery [6] [7]. If aiming at developing intelligent machines for music production, it is evident that these layers must be considered in conjunction. To achieve maximal effectiveness, the mapping model should therefore adapt to the idiosyncrasies of the physical interface and personal abilities of the user. In a similar situation, the natural skills developed through everyday activity could guarantee the user an initial level of expertise, even to people who usually don't have the opportunity to make music [8].

In this direction we conducted a pilot experiment where the user had to mimic the production of simple musical elements. A similar concept appeared in previous literature, under the terms of motor-mimetic sketching, sound-tracing [6], sound "gestureification" [9].

Analogously to those experiments, here as well subjects were demanded to transfer a mental imagery of a perceived acoustical event onto human movement. However, the concurrency of the following aspects tells apart the present account from previous experiments:

- the user's intentions were to be communicated trough a specific task and *control modality* [1]
- the mimicry activity was performed trough a control modality different from the one that produced the sound stimulus (as happens on the contrary for air playing activity [6])
- the user had to rely only on primary feedback [2], i.e. visual, tactile and proprioceptive cues
- the musical stimulus was limited to simple tones, and the user was explicitly asked to address only to sound intensity; this reduction was motivated by the attempt to limit influences of cognitive and cultural aspects raised by melodic and rhythmic developments.
- stimuli perception and control action were tasks sequentially separated

2. METHOD

This experience is intended to explore gestural control for sound production. In particular, we focus on position and velocity control for different sound dynamics, and on gesture coarticulation, a term that defines the "process whereby the properties of a segment are altered due to the influences exerted on it by neighboring segments" [11], referred to with the term *articulation* in the musical domain [12]. In outline, a group of participants was asked to listen to trumpet tones and to mimic their production acting with a device. In the following we describe in detail the components and protocol of the experiment.

[1] for *control modality* we refer to the modality of physical interaction that allows the user to communicate her intentions through a device as in [10]

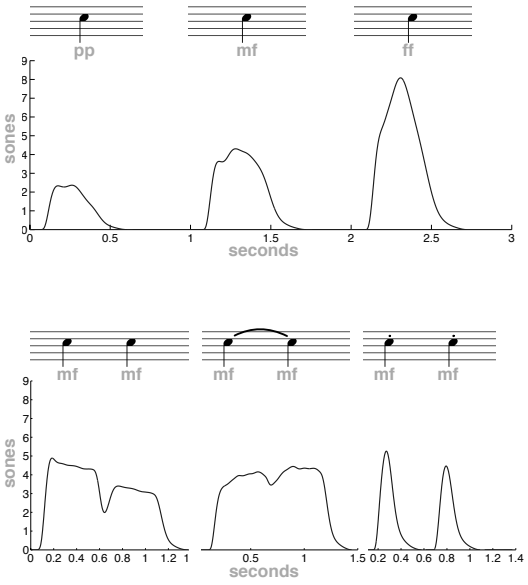


Figure 1. Score notation and loudness profiles computed through Zwicker method [13], for isolated (up) and consecutive (down) notes

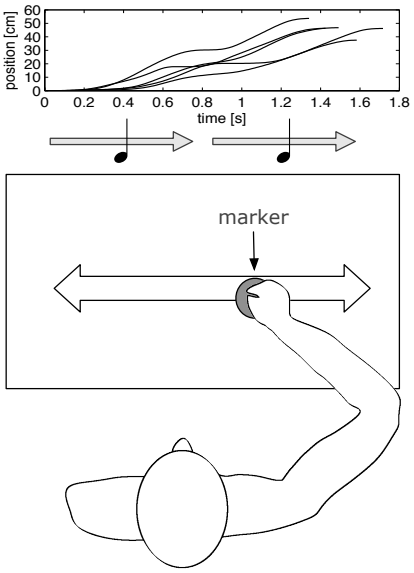


Figure 2. Sketch of the material and configuration setup for the experiment. At top, the displacement over time of the marker, for the mimicry of production of two consecutive tones trough control in velocity

2.1 Stimuli

The acoustic stimuli presented to the subjects consisted of a set of tones produced by a real trumpet performer and recorded during a previous experiment [14]. The audio was presented to the subjects through a headphones set, and the rate of audio amplification was adjusted for each subject in order to assure a neat but comfortable listening. Score notation and loudness profiles of tones are presented in Fig. 1 Isolated notes were presented in triplets of increasing or decreasing loudness, forming a listening stream of ca 2.5s, whereas couple of articulated notes were presented singularly for each condition, with a maximal duration of ca 1.2s.

2.2 Participants

Five subjects took part to the experiment: four males and one female, aged between 25 and 30 years. All subjects were student members or collaborators of the Idmil laboratory, thus involved in research in the musical domain. Part of them were also trained musicians in piano, trumpet and violin performance.

2.3 Material and apparatus

The subjects were seated on a chair in front of a table, and were asked to move a marker leaning on the table by performing planar movement along a line. A sketch of the task setup is shown in Fig. 2 The marker was a sensor of the Polhemus Liberty interface, which tracked its position in time at 120 Hz by means of a custom made driver software developed at Idmil laboratory. The marker was cloth-covered in order to reduce the mechanical resistance of friction of the contact with the table surface. The participants were left free to position their arm and body with respect to the material setup, with the only precaution of finding the most comfortable solution in order to avoid obstruction or limitation of the movement during the performance of the tasks.

2.4 Protocol

The protocol consisted of two stages, each of which involved a different control modality. During the first stage, participants were presented with the series of notes and were asked to mimic the production of the sounds in loudness by relating instantaneous position of the marker with instantaneous value of loudness. Before performing the trials, they were instructed to choose along the line on the table two extreme positions distant 50cm, representatives of zero loudness and of maximal loudness perceived. After listening to each stimulus, subjects had to move the marker from the zero loudness point, across the loudness scale, and back to the origin point, so as to reproduce the loudness profile of the sounds. In the second stage, subjects were asked to relate loudness of sounds with instantaneous velocity of the marker. In this configuration, still position of the marker represented the silence, while an arbitrary maximal linear velocity the maximal loudness perceived. At the beginning of each experiment, the testers were let to familiarize with the task in order to determine the most comfortable movement amplitudes and speeds. Participants were explicitly told to reproduce the loudness profile as accurately as possible, bot in amplitude and temporal variations, and to discard all other auditory attributes. When presented with couples of joined notes, control though velocity was explicitly asked to be performed on the same movement direction, whereas direction was left arbitrary for the tasks with isolated notes.

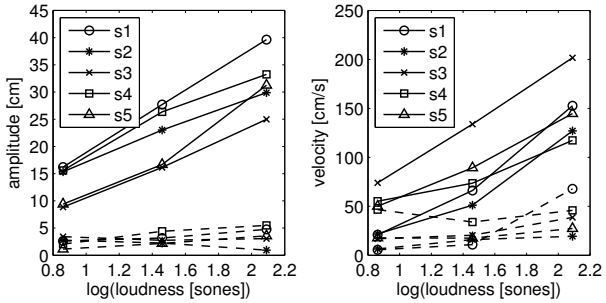


Figure 3. Semi-logarithmic plot of ratio between loudness and maxima in position profiles (left) and velocity profiles (right): mean (solid lines) and standard deviations (dashed line)

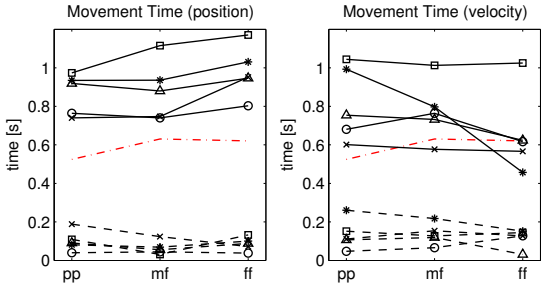


Figure 4. Temporal duration of stimuli and participants' control profiles: : mean (solid lines) and standard deviations (dashed line)

2.5 Analysis

Loudness profiles of the acoustic stimuli were computed with Zwicker method, which makes use of several psychoacoustical principles in order to give an estimate of the average person's impression of the sound intensity for temporally variable sounds [13]. The profiles computed are displayed in Fig. 1. It is worth pointing out that the values of loudness in sones are dependent on an estimation of the real sound pressure level presented to the subjects. These were indeed supposed to have been exposed to an average sound pressure level of 69 db, which lies in the decibel scale as the average comfortable volume for a quiet laboratory setting. The role of possible error in the normalization was investigated and we found that even an error of 20dB would have caused only minor relative difference in our present study.

Each movement trial was manually trimmed from the recordings and examined for amplitude and temporal variations. In a second step, each segment has been resized and rendered in a functional form by means of the FDA Matlab toolbox. Functional conversion with smoothing penalty allowed to represent each movement profile as a continuous and derivable function, and to control smoothness degree of higher order derivatives (which presented irregular spikes due to errors in the sensing and streaming of the capturing system). Control with the generalized cross-validation criterion assured a good compromise between the smoothing effect and the fit to the original curves [15]. In order to have more representative curves for each condition, trial records for the same condition (up to 5) were subsequently aligned by means of landmarks registration. Roughly, this process allows to align features by estimating a strictly increasing nonlinear transformation of time that takes all the times of a given feature into a common value [15].

Indicative measures of shape similarity between normalized curves have been computed by means of the mean square error method described in Ch.8.5 of [15]. This method assured the consideration of an eventual temporal deformation introduced by the registration process. These measures have been computed for the single isolated notes, the single notes embedded in sequence of two notes, and for the entire sequence comprising two notes.

3. RESULTS

The subjects revealed a systematic behavior in the duration and maximal amplitude of control profiles. Fig. 3 displays the maximum values of stimuli loudness versus maximum values of control profiles for the three dynamics conditions. The plot evidences the increase in movement amplitude and peak velocity with loudness, as explicitly requested in the task. For both control modalities, amplitudes generally scale linearly with the logarithm of loudness in sones (a measure unit directly proportional to loudness). The light deviation from a linear trend in the semilog plot for some subjects correlates with singular values in standard deviation or in movement durations. In the case of control by position, higher amplitude for subject 5 in the louder note goes with a greater duration if supposing that he maintained the same amount in velocity. In the case of control by velocity, subjects exhibiting light divergence from linear trend present higher standard deviation or erroneous duration estimation (Fig. 4).

Standard deviation revealed higher for control through velocity, indicating a possible higher difficulty in delivering consistency among trials.

The mean durations of movements together with the ground truth of stimuli events durations for each condition is shown in Fig. 4. Globally the participants tended to overestimate the duration of the event. Comparison of the two plots reveals that this overestimation is dominant for position control. For louder tones, the two modalities reveal a divergent behavior: an increase in duration for control though position and a decrease for control though velocity.

Temporal profiles of stimuli loudness, position, and velocity control for a representative participant are reported in Fig. 5. In plots on second and third columns, the three superposed curves resume the mean of five registered trials for each dynamic condition. Position curves and derivatives conform previous results of human point to point reaching movements, characterized by amplitude invariant symmetric bell-shaped velocity [16]. On the contrary, amplitude invariance does not behold for velocity profiles, whose shape varies to a greater extent between cases. All participants except one manifested a behavior similar to that in figure, transiting from triangular or trapezoidal shape into a bell shape when movement peak velocity exceeds 60cm/s. Subject 3, who was the only participant without

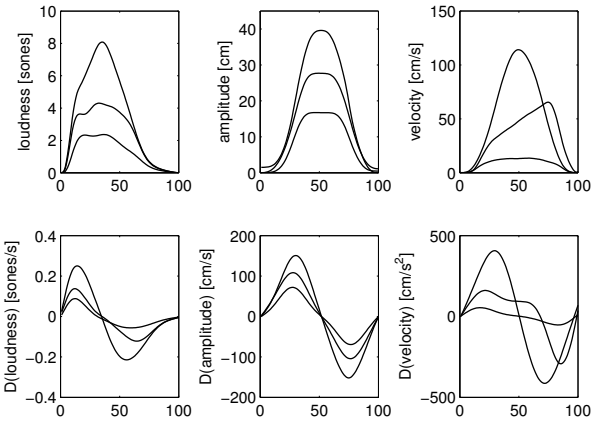


Figure 5. Extract of control profiles for one subject in isolated tones task, for *pp*, *mf*, and *ff* dynamics

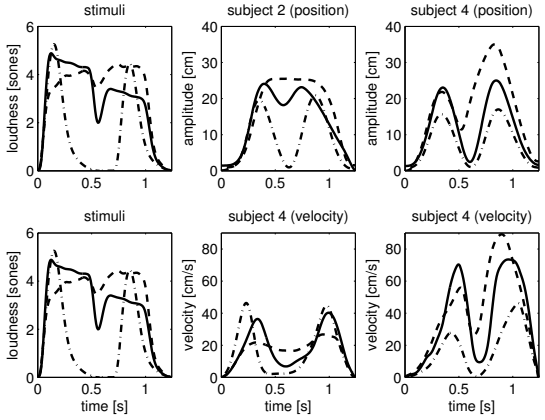


Figure 6. Extract of control profiles for two subjects in consecutive tones task for *non legato* (solid), *legato* (dashed), and *staccato* (dash-dot) articulation. The up row reports profiles in control trough position, the down row profiles in control trough velocity. In first column stimuli loudness, equal for position and velocity tasks

professional musical training, singularly presented similar profiles at all movement amplitudes.

Control profiles for coarticulated notes for two subjects are displayed in Fig 6. The curves reveal distinct method between subjects for the performance of the task, yet preserving the relation between conditions visible in the stimuli profiles. Articulatory degree between the tones reflects in the transient parts between the two strokes, with the level of descend correctly marking the distinction between the three coarticulation strategies. Subject 2 manifested the extreme behavior for position control, performing no descend for the *legato* condition. Subject 4, who was musically trained in violin performance, showed the closest match in profile shapes between the two control modalities.

Indicative values of similarity between curves for each condition are given in Fig. 7, in which higher values represent higher discrepancy between stimuli profile and gesture profile. Results show that control trough position in general performs better then control with velocity in simulating the profile of the stimulus. Moreover, performance required in the task decreases for consecutive notes, both when considering the entire couple segment and when the single embedded note (segmented at minima).

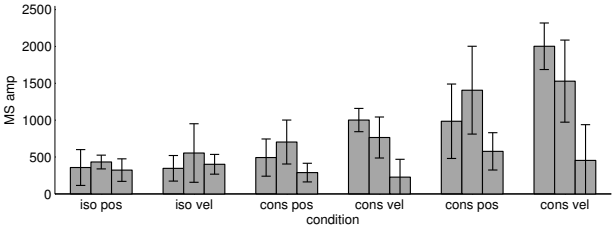


Figure 7. Values of joint amplitude and phase variability as computed by formulas 8.3 and 8.5 in [15]. Higher values represent higher discrepancy between stimulus profile and gesture profile. Values are computed for single note in isolated *pp*, *mf*, and *ff* condition, single note (average) embedded in consecutive *nl*, *l*, and *s* condition, and on entire record for consecutive *nl*, *l*, and *s* condition

4. DISCUSSION

The goal of the present study was to investigate the role of musical dynamics and articulation on human motor control for the mimicry of sound production. Two control modalities were examined, position and velocity.

Both modalities revealed to overestimate the duration of the stimuli, and to scale almost linearly with the logarithm of the loudness perceived. We advance the hypothesis that this scaling could be the outcome of two causes. On a performance level, faster and wider movements could have been automatically reduced by the participant because of arm biomechanical limitations. On a cognitive level, sound loudness could have been mentally associated to movement kinematics - as explicitly demanded in the task - in conjunction with movement effort. Effort on joint torques and muscular activation, which has been shown to scale in amplitude and duration with the movement speed [17] [18] [19], may have participated in reducing the performance of the tasks for wider/faster movements.

The two modalities deviated to some extent in variability. If we consider variability in-between trials as an indicator of difficulty, Fig. 3 and Fig. 4 suggest that control trough velocity could be a harder task than control by position. A substantial reliance on visual over kinesthetic feedback for position control could be the cause for a more consistence performance. Support of this hypothesis is given by the fact that participants with developed acuteness for arm velocity profiles (typical of violin players, alike subject 4) delivered comparable results between the two modalities.

The two control modalities differentiate also in terms of profile shape (Fig. 5). Position control manifested over all dynamics conditions a bell-shaped velocity profile typical of point to point movements, whose velocity profile has been largely proved to be invariant to duration, distance, and peak velocity [17] [21] [16] [22]. Control in velocity, on the contrary, did not show the same invariant

property. By comparing our profiles (Fig. 5) with theoretical accounts that addressed the modeling of movement in terms of effort minimization [23] [24], we can advance the hypothesis that a change in the motor control strategy adopted in the mimicry took place along with the change in stimuli dynamics.

To our knowledge, no study on human pointing movements proposed requirements similar to our experiment. Literature on pointing movement tasks usually refers to Fitt's law for a quantitative description between movement time and amplitude. However, Fitt's model grounds on the condition of self-paced movements, that is movements in which the execution time behaves as a by-product of the speed-accuracy trade off - participants are required to move as fast and as accurate as possible. This model however has proved to fail in describing tasks where subjects are required to move at a specified time [20]. In this experiment the movement task is based both on temporal and on spatial constraints. Participants had to assure maximal accuracy both in the end point positions (intensity levels) and in the movement timing (duration and temporal profile). In the musical domain, similar conditions form the requisites for the performance of the violin, for which bowing techniques have been investigated in terms of motor control strategies in [25].

The comparison between profiles for isolated and consecutive notes revealed that the sequential ordering of gesture units do not resolve in simple temporal sequencing, but entails a structural change to its constituents which affects the all sequence. Velocity profiles for velocity based control, indeed, converted to bell-shaped in all participants even for peak velocities under 60 cm/s, contrary to the case of isolated notes, thus revealing that in terms of motor control, a different strategy may be in use.

Whether the mimicry of two notes should be considered as the repetition of two discrete tasks or as a per-se unary rhythmic task is an open question, which still puzzles general motor control research [26]. Brain imaging studies revealed that the performance of rhythmic movements involve different brain areas then when performing discrete movements [27], giving support to the idea that rhythmic movements cannot be considered as the concatenation of discrete movements. In the present experiment, stimuli quantity was limited to two elements in order to prevent the emergence of frequency or pace effects. However, change in profiles and different values of similarity between gestures and stimuli (Fig. 7) lead us to suppose that to some degree a change in behavior took place.

5. CONCLUSION AND FUTURE DIRECTION

In this paper we have presented the description and results of a pilot experiment in which participants mimicked with different control modalities the production of sound stimuli. In the broader context of human-computer interaction, our experiment sets, to put it as Buxton [28], on *pragmatism*, in that it considers the interdependence of transducer or control modality with the visual and kinesthetic skills engaged in the interaction. Idiosyncrasies between modalities and between users, both in values and temporal profiles, indicate that a mapping strategy capable to adapt to the control channel and to the user natural skills could accelerate the "process whereby novices begin to perform like experts" [28]. On a higher level, recognition of coarticulation effects may help in extracting semantic cues on the embedding of gesture units in human-computer phrasal dialogue.

For a future case study, we envision to improve some aspects of the experiment. First, the substitution of headphones with loudspeakers will help in the monitoring of the effective acoustical intensity delivered to the subjects. Secondly, we foresee to use trumpet tones synthetically produced by a physical modeling synthesis software [2] as acoustical stimuli. Synthesized material will permit to avoid dissimilarity between tones (see Fig. 1) caused by inconsistency in the trumpet player performance, consequently excluding the presence of uncontrolled external factors that could interfere in the mimicry task. We are currently working on the trumpet model in order to augment tongue and airflow interaction for a more realistic simulation of articulation techniques.

6. AKNOWLEDGEMENTS

This work was partially supported by Short-Term Scientific Mission program of the COST Action IC0601 on Sonic Interaction Design

7. REFERENCES

[1] B. Shackel, "Human factors and usability," pp. 27–41, 1990.

[2] R. Vertegaal, T. Ungvary, and M. Kieslinger, "Towards a musician's cockpit: Transducers, feedback and musical function," in *Proceedings of the International Computer Music Conference*, pp. 308–310, 1996.

[3] M. M. Wanderley, J. Viollet, F. Isart, and X. Rodet, "On the choice of transducer technologies for specific musical functions," in *Proceedings of the International Computer Music Conference*, 2000.

[4] M. M. Wanderley and N. Orio, "Evaluation of input devices for musical expression: Borrowing tools from hci," *Comput. Music J.*, vol. 26, no. 3, pp. 62–76, 2002.

[5] B. Caramiaux, F. Bevilacqua, and N. Schnell, "Towards a gesture-sound cross-modal analysis." Lecture Notes in Computer Science. Springer-Verlag (to appear), 2008.

[6] R. I. Gody, E. Haga, and A. R. Jensenius, "Playing air instruments: Mimicry of sound-producing gestures by novices and experts," in *Proceedings of the 6th International Gesture Workshop* (J.-F. K. Sylvie Gibet, Nicolas Courty, ed.), 2006.

[7] M. Leman, *Embodied Music Cognition and Mediation Technology*. Cambridge: MIT, 2008.

[2] http://forumnet.ircam.fr/701.html

[8] A. Boulanger, "Expressive gesture controller for an individual with quadriplegia," in *Proceedings of the 10th Ubicomp 2008 Adjunct Programs*, pp. 113–116, 2008.

[9] N. Schnell, "Collaboration on sound gesturefication." SID STSM Report, 2008.

[10] Q. Wang, S. Harada, T. Hsieh, and A. Paepcke, "Visual interface and control modality: An experiment about fast photo browsing on mobile devices," in *Proceedings of Human Computer Interaction INTERACT*, Lecture Notes in Computer Science, Springer Berlin, 2005.

[11] R. Hammarberg, "The metaphysics of coarticulation," *Journal of Phonetics*, vol. 4, pp. 353–363, 1976.

[12] R. J. Jackson, *Performance Practice: A Dictionary-Guide For Musicians*. New York: Routledge, 2005.

[13] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*. Springer-Verlag New York, Inc., 2006.

[14] T. Bianco, V. Freour, N. Rasamimanana, F. Bevilacqua, and R. Caussé, "On gestural variation and coarticulation effects in sound control," *Lecture Notes in Computer Science [to appear]*.

[15] J. Ramsay, G. Hooker, and S. Graves, *Functional Data Analysis with R and MATLAB*. Springer, 2009.

[16] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[17] C. Gielen, K. van den Oosten, and F. Pull ter Gunne, "Relation between emg activation patterns and kinematic properties of aimed arm movements," *Journal of Motor Behavior*, vol. 17, no. 4, pp. 421–42, 1985.

[18] Hollerback and T. Flash, "Dynamic interactions between limb segments during planar arm movement," *Biological Cybernetics*, vol. 44, no. 1, pp. 67–77, 1982.

[19] Flanders and Herrmann, "Two components of muscle activation: scaling with the speed of arm movement," *Journal of Neurophysiology*, vol. 67, no. 4, pp. 931–943, 1992.

[20] R. A. Schmidt, H. N. Zelaznik, B. Hawkins, J. S. Frank, and J. T. Quinn, "Motor-output variability: A theory for the accuracy of rapid motor acts," *Psychological Review*, vol. 86, pp. 415–451, 1979.

[21] C. G. Atkeson and J. M. Hollerback, "Kinematic features of unrestrained arm movements," *Journal of Neuroscience*, vol. 5, pp. 2318–2330, 1985.

[22] J. Soechting and F. Lacquaniti, "Invariant characteristics of a pointing movement in man," *Journal of Neuroscience*, vol. 1, pp. 710–720, 1981.

[23] W. Nelson, "Physical principles for economies of skilled movements," *Biological Cybernetics*, vol. 46, no. 2, pp. 135–147, 1983.

[24] N. Hogan, "An organizing principle for a class of voluntary movements," *Journal of Neuroscience*, vol. 4, no. 11, pp. 2745–2754, 1984.

[25] N. Rasamimanana and F. Bevilacqua, "Effort-based analysis of bowing movements: evidence of anticipation effects," *The Journal of New Music Research*, vol. 37, no. 4, pp. 339 – 351, 2008.

[26] N. Hogan and D. Sternad, "On rhythmic and discrete movements: reflections, definitions and implications for motor control," *Experimental Brain Research*, vol. 181, no. 1, pp. 13–30, 2007.

[27] S. Schaal, D. Sternad, R. Osu, and M. Kawato, "Rhythmic arm movement is not discrete," *Nature Neuroscience*, vol. 7, no. 10, pp. 1136–1143, 2004.

[28] W. Buxton, "Chunking and phrasing and the design of human-computer dialogues," pp. 475–480, 1986.

HUBS AND ORPHANS - AN EXPLORATIVE APPROACH

Martin Gasser
Austrian Research Institute
for Artificial Intelligence (OFAI)
martin.gasser@ofai.at

Arthur Flexer
Austrian Research Institute
for Artificial Intelligence (OFAI)
arthur.flexer@ofai.at

Dominik Schnitzer
Austrian Research Institute
for Artificial Intelligence (OFAI)
dominik.schnitzer@ofai.at

ABSTRACT

In audio based music similarity, a well known effect is the existence of hubs, i.e. songs which appear similar to many other songs without showing any meaningful perceptual similarity. We show that this effect depends on the homogeneity of the samples under consideration. We compare three small sound collections (consisting of polyphonic music, environmental sounds, and samples of individual musical instruments) with regard to their hubness. We find that the collection consisting of cleanly recorded musical instruments produces the smallest hubs, wheras hubness increases with inhomogeneity of the audio signals. We also investigate how well the three data sets can be mapped into a 2D visualization space by a dimensionality reduction algorithm based on Multidimensional Scaling.

1. INTRODUCTION

One of the central goals in Music Information Retrieval is the computation of audio similarity. Proper modeling of audio similarity enables a whole range of applications: music classification/recommendation, content-based search, etc. The de facto standard approach to computation of audio similarity is timbre similarity based on parameterization of audio using Mel Frequency Cepstral Coefficients (MFCCs) plus Gaussian mixtures as statistical modeling (see Sec. 3.1). However, it is also an established fact that this approach suffers from the so-called hub problem [1]: sound samples which are, according to the audio similarity function, similar to very many other sound samples without showing any meaningful perceptual similarity to them. The hub problem of course interferes with all applications of audio similarity: hub samples keep appearing unwontedly often in recommendations, they degrade classification performance, etc.

Although the phenomenon of hubs is not yet fully understood, a number of results already exist. Aucouturier and Pachet [2] established that hubs are distributed along a scale-free distribution, i.e. non-hub samples are extremely common and large hubs are extremely rare. This is true for MFCCs modelled with different kinds of Gaussian mixtures as well as Hidden Markov Models, irrespective whether parametric Kullback-Leibler divergence or non-parametric histograms plus Euclidean distances are used for computation of similarity. But is also true that hubness is not the property of a sound sample per se since non-parametric and parametric approaches produce very different hubs. It has also been noted that audio recorded from urban soundscapes, different from polyphonic music, does not produce hubs [3] since its spectral content seems to be more homogeneous and therefore probably easier to model. Direct interference with the Gaussian models during or after learning has also been tried (e.g. homogenization of model variances) although with mixed results. Whereas some authors report an increase in hubness [2], others observed the opposite [4]. Using a Hierarchical Dirichlet Process instead of Gaussians for modeling MFCCs seems to avoid the hub problem altogether [5].

Our main interest in this paper is to explore whether the hub problem also exists in data bases of audio samples (e.g. short recordings of individual notes played on individual instruments) rather than data bases of whole songs. We are also interested in finding out how hubs influence lower dimensional projections of audio data bases. Such projections have been very popular for enabling interactive visualizations of data bases of songs [6–8] as well as samples [9, 10]. Despite the popularity of these interfaces based on lower dimensional projections, it has not yet been clarified how hubs influence and possibly impair these visualizations.

Our contribution to the understanding of the hub problem and its consequences are twofold: (i) We support the assumption that hubness is less predominant in collections of environmental sound textures [3] than in music collections, and show that it is even weaker in musical instrument databases and (ii) we show that hubness is related to the cluster structure of data by inspecting Multidimensional Scaling projections (i.e. hubs tend to lie in the centers of clusters, the more clusters can be found in a data set, the smaller are the hubs).

2. DATA

In this research, we used three sound collections of $size = 1000$. However, we assume that the results remain valid for bigger databases as well.

2.1 Music collection

We used a subset of a data base comprised of the music of an Austrian music portal. The FM4 Soundpark is an in-

Category	Number of samples
Foreign Towns & Countrysides	10
Nature Ambiences	22
Birds and Animals	73
Wind	38
Water	49
Rain	33
Planes and Trains	52
Cars	99
Traffic	49
Guns	91
Crashes&Impacts	19
Sports and Boats	85
Towns	21
General Ambiences	359

Table 1. Structure of environmental sounds collection

Instrument	Number of samples
Strings Ensemble	185
Double Bass Ensemble	135
Clarinet	191
Tuba	133
Drums	200
Flute	56
Horn	100

Table 2. Structure of musical instruments collection

ternet platform [1] of the Austrian public radio station FM4. This internet platform allows artists to present their music free of any cost in the WWW. All interested parties can download this music free of any charge. This music collection contains about 10000 songs and is organized in a rather coarse genre taxonomy. The artists themselves choose which of the $G_M = 6$ genre labels "Hip Hop, Reggae, Funk, Electronic, Pop and Rock" best describe their music. The artists are allowed to choose one or two of the genre labels. We use a data base of $size = 1000$ songs for our experiments.

2.2 Environmental sample collection

The collection of environmental sounds consists of subsets of the commercially available sound effects libraries *Sound Ideas* [2] and *Hollywood Edge* [3].

2.3 Collection of musical instruments

The musical instruments collection consists of a subset of a commercially available sample collection for professional composers. Tab. 2 gives an overview of the structure of our evaluation dataset.

[1] http://www.soundpark.at
[2] http://www.sound-ideas.com/
[3] http://www.hollywoodedge.com/

3. METHODS

In this paper we use Single Gaussian (G1) distributions of Mel Frequency Cepstral Coefficients (MFCCs) to model the spectral content of individual sound samples; based on the bag-of-frames approach, those models represent the average spectral envelope and covariances between the individual coefficients, corresponding to the specific "sound" or "timbre" of a sample. By comparing the statistical models using the symmetric Kullback-Leibler divergence, we calculate a dissimilarity measure between samples.

For the visualization mapping, we arrange sound samples in 2D space, such that the distances between the 2D locations approximate the acoustic distances. To construct the 2D arrangement, we use Multidimensional Scaling [11].

3.1 Mel Frequency Cepstral Coefficients and Single Gaussians (G1)

We use the following approach to compute acoustic similarity. For a given collection of sound samples, it consists of the following steps:

1. for each sample, compute MFCCs for short overlapping frames
2. train a single Gaussian (G1) to model each of the samples
3. compute a distance matrix M_{G1} between all samples using the symmetrized Kullback-Leibler divergence between respective G1 models

The sound samples are resampled to 22050Hz and mixed down to mono audio signals. Then, we divide the raw audio data into overlapping frames of short duration and use Mel Frequency Cepstral Coefficients (MFCC) to represent the spectrum of each frame. MFCCs are a perceptually meaningful and spectrally smoothed parameterization of audio signals and a standard technique for computation of spectral similarity in music analysis (see e.g. [12]). The frame size for computation of MFCCs for our experiments was $23.2ms$ (512 samples), we used a hop size of $11.6ms$ (256 samples). We used the first $d = 20$ MFCCs for all experiments.

A single Gaussian (G1) with full covariance represents the MFCCs of each sample [13]. For two single Gaussians, $p(x) = \mathcal{N}(x; \mu_p, \Sigma_p)$ and $q(x) = \mathcal{N}(x; \mu_q, \Sigma_q)$, the closed form of the Kullback-Leibler divergence is defined as [14]:

$$KL_N(p\|q) = \frac{1}{2}\left(\log\left(\frac{\det(\Sigma_p)}{\det(\Sigma_q)}\right) + Tr\left(\Sigma_p^{-1}\Sigma_q\right) + (\mu_p - \mu_q)'\,\Sigma_p^{-1}(\mu_q - \mu_p) - d\right) \quad (1)$$

where $Tr(M)$ denotes the trace of the matrix M, $Tr(M) = \Sigma_{i=1..n} m_{i,i}$. The divergence is symmetrized by computing:

$$KL_{sym} = \frac{KL_N(p\|q) + KL_N(q\|p)}{2} \quad (2)$$

3.2 Visualization algorithm

It is well known that the symmetric KL divergence between Gaussians does not satisfy the requirements to a proper distance measure [15]. However, it is possible to embed data items in a Euclidean space, such that the Euclidean distances between pairs approximate the original data distances as closely as possible. Such techniques are generally termed *Multidimensional Scaling* [16]. One possible approach to solve the Multidimensional Scaling problem are spring models [17, 18], physically inspired models of spring-connected nodes aiming at finding a node placement that minimizes the cumulative deflection from the springs' resting states. Mapping distances in feature space to spring lengths, a spring model solves the MDS problem. Implicitly, this implements a gradient-descent based solution algorithm, where a placement of the nodes that minimizes the overall stress (the deflection from the springs' resting state) is constructed. Eq. 3 gives a measure for the normalized stress in an MDS mapping.

$$S = \frac{\Sigma_{i<j}(d_{ac}(i,j) - d_{lo}(i,j))^2}{\Sigma_{i<j} d_{lo}(i,j)} \quad (3)$$

(d_{ac}: acoustic distance, d_{lo}: distance in low dimensional space, i, j: data points)

4. RESULTS

4.1 Hubs in sample databases

As a measure of the hubness of a given sample, we use the so-called n-occurrence [2], i.e. the number of times the sample occurs in the first n nearest neighbors of all the other samples in the data base. Please note that the mean n-occurrence across all samples in a data base is equal to n. Any n-occurrence significantly bigger than n therefore indicates existence of a hub. For every sample in the data bases, we computed the first n nearest neighbors. The first n nearest neighbors are the n samples with minimum Kullback-Leibler divergence (Equ. 2) to the query sample.

The results given in Tab. 3 show results calculated from the three data sets. We give the number of nearest neighbors n, the absolute number of the maximum n-occurrence $maxhub$ (i.e. the biggest hub), the percentage of samples in whose nearest neighbor lists this biggest hub can be found $maxhub\% = maxhub/size$ and the percentage of hubs $hub3\%$ (i.e. the percentage of samples of which the n-occurrence is more than three times n).

When looking at the results, it should be immediately clear that music collections are more prone to hubness than collections of environmental textures or musical instruments. This can be intuitively justified because the spectral structure of music is much less homogeneous than that of sound textures or even individual musical instruments. Thus, it is quite likely that a given musical piece is similar to a higher-than-average number of other pieces.

4.2 MDS mappings of the data sets

Fig. 1(a), 1(b), and 1(c) show 2D embeddings of the music, textures, and instruments data sets, respectively. We ran 1500 iterations of the MDS algorithm on the data. Additionally to the location of the samples, we also plotted the 50 largest hubs (samples that appear most often in the nearest neighbor lists of all other samples) and the smallest orphans (samples that rarely appear in the nearest neighbor lists of all other samples). Hubs are marked with a green dot, orphans with a red 'X'-sign.

data set	n	maxhub	maxhub%	$hub3\%$
MUSIC	50	358	35.80	5.20
TEXTURES	50	190	19.04	0.90
INST	50	172	17.18	0.29

Table 3. Hub analysis results for the three data sets

data set	stress
MUSIC	0.0327
TEXTURES	0.0434
INST	0.0556

Table 4. Stress values after 1500 MDS iterations

The music data in fig. 1(a) seems to contain only one cluster and all hubs are located within the center of this cluster. Orphans are located at the border of the point cloud.

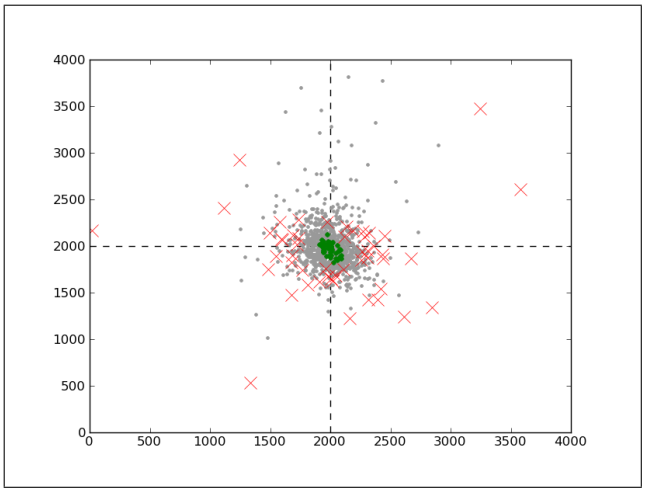
Fig. 1(b) and fig. 1(c) show a different picture. Here we can clearly see the existence of a more complex clustering in the data (e.g., sounds of engines vs. wind/sea textures, flutes vs. double bass) and the hubs are distributed more evenly over the clusters. We can also see that orphans do not strictly lie at the border of the point cloud anymore. Since MDS aims at minimizing the sum of squared differences between high- and low-dimensional distances, it is clear that in the music data set, hubs (i.e. samples with small distance to a large number of other samples) should go to the center of the point cloud, whereas orphans are pushed to the borders.

To measure the performance of MDS, we evaluate the stress formula 3 for all three data sets. Tab. 4 shows the stress values after 1500 iterations of the gradient descent-based MDS algorithm. Apparently, it is more difficult for the algorithm to cope with data sets containing multiple clusters, whereas the data set containing one cluster yields the lowest stress value, although it also contains the largest hubs.

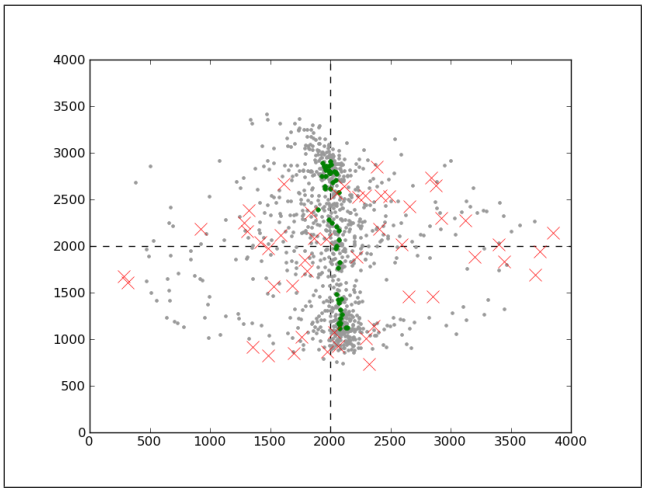
5. CONCLUSION & FUTURE WORK

We have shown that music collections are indeed more prone to the *hubness* problem than collections of sound textures or musical instruments. We suppose that this is due to the high degree of inhomogeneity in individual musical pieces, whereas environmental sounds or individual instruments have a relatively stationary spectral structure.

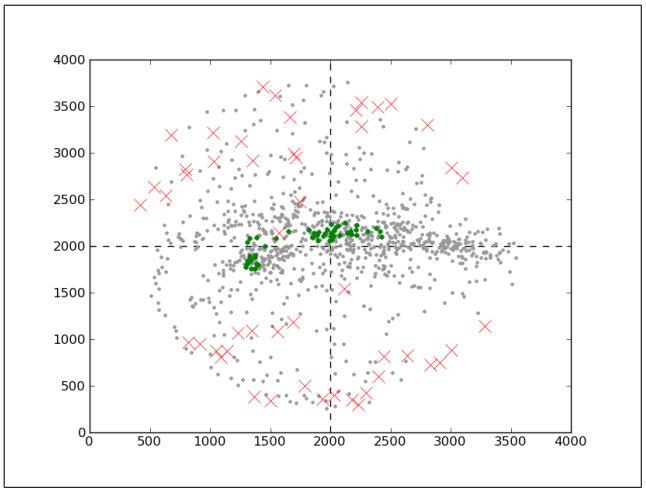
We have also shown that hubs are generally located inside clusters of data, thus, the more clusters are present in the data, the lower is the degree of hubness, since hubs are distributed across clusters.



(a) Music



(b) Textures



(c) Instruments

Figure 1. Distribution of hubs and orphans in MDS projections of the data sets.

For sound textures and samples of musical instruments, the G1 similarity measure produced far fewer hubs than for the music collection; the MDS visualizations of textures and instruments also reflected the structure of the data sets visually, which could be an important argument for successful navigation inside sample libraries.

We have also developed a demonstration application that can be used to interactively explore the data sets we used by zooming/panning in the MDS embeddings and by playing back individual audio samples. It can be used to easily retrieve similar-sounding material from sample libraries without using any metadata.

Next steps in our research will include experiments with other features that also reflect time-dependent properties of the signal and the development of more advanced ways to visualize data in multiple feature spaces.

6. ACKNOWLEDGEMENTS

This research is supported by the Austrian Science Fund (FWF, P21247) and the Vienna Science and Technology Fund (WWTF, project Audiominer)

7. REFERENCES

[1] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky?," *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, 2004.

[2] J.-J. Aucouturier and F. Pachet, "A scale-free distribution of false positives for a large class of audio similarity measures," *Pattern Recognition*, vol. 41, no. 1, pp. 272–284, 2007.

[3] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[4] M. Godfrey and P. Chordia, "Hubs and homogeneity: Improving content-based music modeling," in *Proceedings of the 9th International Conference on Music Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, (Philadelphia, USA), 2008.

[5] M. Hoffman, D. Blei, and P. Cook, "Content-based musical similarity computation using the hierarchical dirichlet process," in *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, (Philiadelphia, USA), 2008.

[6] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," in *Proceedings of the 10th ACM International Conference on Multimedia*, (Juan les Pins, France), pp. 570–579, 2002.

[7] P. Knees, M. Schedl, T. Pohle, and G. Widmer, "Exploring music collections in virtual landscapes," *IEEE MultiMedia*, vol. 14(3), pp. 46–54, 2007.

[8] M. Gasser and A. Flexer, "FM4 soundpark audio-based music recommendation in everyday use," in *Proceedings of the 6th Sound and Music Computing Computing Conference (SMC 09)*, (Porto, Portugal), 2009.

[9] E. Pampalk, P. Hlavac, and P. Herrera, "Hierarchical organization and visualization of drum sample libraries," in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFx'04)*, (Naples, Italy), October 5-8 2004.

[10] S. Heise, M. Hlatky, and J. Loviscach, "Aurally and Visually Enhanced Audio Search With Soundtorch," in *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, (New York, NY, USA), pp. 3241–3246, ACM, 2009.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[12] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *Proceedings of the 1st International Conference on Music Information Retrieval*, (Plymouth, Massachusetts), 2000.

[13] M. Mandel and D. Ellis, "Song-level features and support vector machines for music classification," in *Proceedings of the 6th International Conference on Music Information Retrieval*, (London, United Kingdom), 2005.

[14] W. Penny, "Kullback-leibler divergences of normal, gamma, dirichlet and wishart densities," tech. rep., Wellcome Department of Cognitive Neurology, 2001.

[15] E. Pampalk, *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, March 2006.

[16] M. Cox and M. Cox, *Multidimensional Scaling*. Chapman and Hall, 2001.

[17] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[18] A. Morrison, G. Ross, and M. Chalmers, "Fast multidimensional scaling through sampling, springs and interpolation," *Information Visualization*, vol. 2, no. 1, pp. 68–77, 2003.

RESTORATION OF AUDIO DOCUMENTS WITH LOW SNR: A NMF PARAMETER ESTIMATION AND PERCEPTUALLY MOTIVATED BAYESIAN SUPPRESSION RULE

Giuseppe Cabras
Dep. of Electrical, Management and Mechanical Engineering
University of Udine, Italy
giuseppe.cabras@uniud.it

Sergio Canazza
Sound and Music Computing Group,
Dep. of Information Engineering
University of Padova, Italy
canazza@dei.unipd.it

Pier Luca Montessoro, Roberto Rinaldo
Dep. of Electrical, Management and Mechanical Engineering
University of Udine, Italy
{montessoro, rinaldo}@uniud.it

ABSTRACT

In the field of audio restoration, the most popular method is the Short Time Spectral Attenuation (STSA). Although this method reduces the noise and improves the SNR, it mostly tends to introduce signal distortion and a residual noise called musical noise (a tonal, random, isolated, time-varying noise). This work presents a new audio restoration algorithm based on Non-negative Matrix Factorization (NMF) with a noise suppression rule that introduce the masking phenomenon of the human hearing to calculate a noise masking threshold from the estimated target source. Extensive test with PESQ measure at low SNR (i.e. < 10dB) show that the method does not introduce musical noise and permits to control the trade-off between undesired component suppression and source attenuation. In particular, we show that NMF is a suitable technique to extract the clean audio signal from undesired non stationary noise in a monaural recording of ethnic music. Moreover, we carry out a listening test in order to compare NMF with the state of the art audio restoration framework using the EBU MUSHRA test method. The encouraging results obtained with this methodology in the presented case study support their applicability in several fields of audio restoration.

1. INTRODUCTION

The ethnic-musical heritage – often the only testimonial of past oral cultures – is in danger of disappearing: the audio documents were usually recorded in non-professional carriers by means of amateur recording system. Thus, for their appropriate fruition and/or for a suitable use of Music Information Retrieval techniques it is necessary to process the signals by means of audio restoration algorithms.

Different strategies can be adopted in a combined way with audio restoration algorithms, in accordance with the final purposes of the access copy:

- Documental approach: in this case, the de-noising algorithms only concern the cases in which the internal evidence of the degradation is unquestionable, without going beyond the technological level of that time.
- Aesthetical approach: it pursues a sound quality that matches the actual user's expectations (for both new commercial editions and to arrange the signal before the use of MIR techniques).
- Sociological approach: it has the purpose of obtaining a historical reconstruction of the recording as it was listened to at the time (see Storm, Type I [1]).
- Reconstructive approach: it has the objective of preserving the intention of the author (see Storm, Type II [1])

In order to reach one or more of the above aims, it is necessary to have at disposal several audio restoration instruments (often in the same audio document there are corruptions with different physical characteristics, that can be attenuated with different de-noise filters). The audio restoration algorithms can be divided into three categories [2]:

1. frequency-domain methods, such as various forms of non-casual Wiener filtering or spectral subtraction schemes and recent algorithms that attempt to incorporate knowledge of the human auditory system; these methods use little a priori information;
2. time-domain restoration by signal models such as Extended Kalman Filtering (EKF): in these methods a lot of a priori information is required in order to estimate the statistical description of the audio events;
3. restoration by source models: only a priori information is used.

The advantage of frequency-domain methods is that they are straightforward and easy to implement. However, the limitations are as follows: musical noise (short sinusoids randomly distributed over time and frequency) is unavoidable; the results depend on a good noise estimation. Restoration by source model is limited to very few cases (e.g. only monophonic recordings) and it is not generalizable. The EKF is able, in principle, to simultaneously solve the problems of filtering, parameter tracking and elimination of the

outliers, but it is very sensitive to parameter setting and ineffective where the Signal-to-Noise Ratio (SNR) is very low (< 10dB), as happen in many ethnic music audio documents.

This work presents a new audio restoration method – that fall within the first category – based on Non-negative Matrix Factorization (NMF), an emerging new technique in the blind extraction of signals recorded in a variety of different fields. The application of NMF to the analysis of monaural recordings is relatively recent. We show that NMF is a suitable technique to extract the clean audio signal from undesired non stationary noise in a monaural recording of ethnic music. More specifically, based on finding by Wolfe and Godsill [3], we develop a perceptually motivated distortion measure as a generalization of the Minimum Mean Square Error (MMSE) cost function that incorporates the masking threshold. Moreover, we carry out a listening test in order to compare NMF with the state of the art audio restoration framework using the EBU MUSHRA test method.

A recent approach to separate an acoustic source is provided by Non-negative Matrix Factorization (NMF). The basic idea is that we can obtain a meaningful *part-based* factor decomposition [4] from a data observation (e.g., the monaural recording) by the only constrain of non-negativity and sparsity, since no cancellation of factors can occur and only additive combinations are permitted. The use of sparse code can favor a factorization where only a few dictionary elements are used to model the source, introducing an ℓ_1 norm penalty term on the coefficients of the code matrix, which explicitly enforces sparseness [5]. However, a further non trivial step is needed to assign the decomposed parts to the source of interest (e.g., the original audio signal) to discard the interference source (e.g., the corrupting noise). The proposed approach tries to solve this problem with a solution based on an extended Non-negative Matrix Factorization algorithm and prior knowledge on interference. In addition, our approach reduces both distortion and perceptually annoying musical noise by taking into account the masking phenomenon of the human hearing, in order to calculate a noise masking threshold from the estimated target source.

We apply this method to improve the quality of noisy recordings of ethnic music on Shellac 78 rpm phonographic discs. The Shellac disc is a common audio mechanical carrier, where the audio information is recorded by means of a groove cut into the surface by a stylus modulated by the sound, either directly in the case of acoustic recordings or by electronic amplifiers. There are more than 1,000,000 Shellac discs in the worldwide audio archives containing music never re-recorded (R&B, Jazz, Ethnic, Western classical, etc.).

The rest of this paper is organized as follows. Sec. 2 details the proposed audio restoration method: in particular, Sec. 2.5 introduces perceptually motivated Bayesian suppression rules used. In order to validate the system, we carry out a listening test – using ethnic music audio documents – in order to compare NMF with the state of the art audio restoration framework using the EBU MUSHRA test method (Sec. 3). Final conclusions are drawn in Sec. 4.

2. AUDIO ENHANCEMENT FRAMEWORK

The objective of the proposed method is to estimate the undesired components, or interference, $n(t)$ and the source of interest, or target, $s(t)$ directly from the observable data mix (i.e. in the time domain), with the minimum *a priori* knowledge. We assume that saturation effects are absent in the mixed observable signal $x(t)$, that can be expressed as:

$$x(t) = s(t) + n(t) \tag{1}$$

We assume that $s(t)$ and $n(t)$ are uncorrelated. This extends linearity in the power spectral domain, and let us to transform the data in a non-negative representation suitable for NMF processing:

$$|X(t,f)|^2 = |S(t,f)|^2 + |N(t,f)|^2 \tag{2}$$

where the observable signal $x(t)$ is transformed in a time-frequency representation $X(t,f)$. Our method is shown in Fig. 1 and functional modules are discussed in the next subsections.

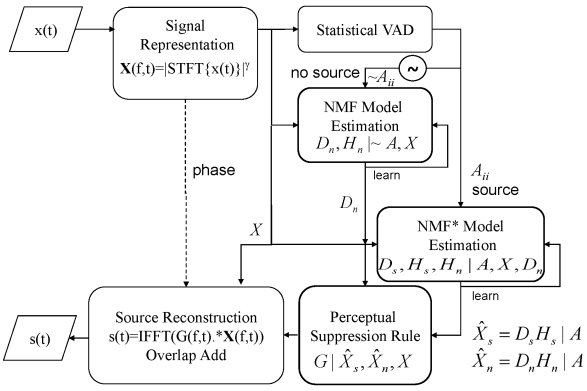


Figure 1. *General scheme of the proposed audio enhancement framework.*

2.1 Signal Representation

A common technique to manipulate audio signals consists of transforming the time-varying observed signal in a time-frequency representation (by means a Short Time Fourier Transform – STFT – analysis) which shows the signal energy variation along time elements (frames) and frequency elements (bins), thus providing a non-negative matrix representation. In the following, we represent the signal in the time-log frequency domain as an element-wise exponentiated STFT:

$$X = |STFT\{x(t)\}|^{\gamma} \tag{3}$$

The linearity expressed by Eq. 2 applies also to Eq. 3 when $\gamma = 2$, but wide experimentation shows that γ is an important parameter to NMF performance. In particular, it turns out that $\gamma = 2$ is a bad choice for component separation, while an optimal choice is $\gamma = 0.67$, which corresponds to the cube root compression of power STFT.

Surprisingly, this is consistent with Stevens' Power Law exponent for the perceived loudness of a sound pressure of 3 kHz tone stimulus. Moreover, Stevens' Power Law was used to model cochlear non-linearities [6] and intensity to loudness conversion in Perceptual Linear Predictive (PLP) speech analysis [7]. More recently, Plourde and Champagne integrated the cochlear compressive nonlinearity in a Bayesian Short Time Spectral Attenuation (STSA) estimation for speech enhancement [8]. This curious coincidence about the exponent value, suggests to follow a perceptually motivated approach to audio de-noising, as we explain in Sec. 2.5.

2.2 Voice Activity Detection

A Voice Activity Detector (VAD) is widely used as a component of speech enhancement methods to update the noise spectrum frame by frame. In our implementation, a statistical-model based VAD [9] is used to construct two diagonal binary square matrices:

$$A(t,t) = \begin{cases} 1, & \text{if target source is present in frame t} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

and its complementary $\bar{A}(t,t)$.

This allows us to train the undesired components dictionary, computing NMF on the signal:

$$Z(f,t) = X(f,t)\bar{A}(t,t) \quad (5)$$

during target-absent periods, and then separate the target components dictionary, computing a modified NMF^* on the signal:

$$Y(f,t) = X(f,t)A(t,t) \quad (6)$$

during target-present periods. Assuming that the target and the undesired component are additive (as stated in Eq. 1), the VAD module has to decide, for each frame t, in favor of one of the two hypotheses:

$$H_0 : X_f = N_f : \quad \text{target source absent,} \quad (7)$$
$$H_1 : X_f = S_f + N_f : \quad \text{target source present.} \quad (8)$$

2.3 Undesired component training

During training stage, we assume availability of some target-absent frames, computed applying a VAD to the observable signal $X(f,t)$; the resulting signal $Z(f,t)$ of Eq. 5 is equivalent to $X(f,t)$, with target-present frame suppressed. Applying a Regularized Euclidean NMF to $Z(f,t)$, we obtain the strictly positive dictionary $D_n(f,k)$ and sparse code $H_n(k,f)$ matrices, where k is the number of user defined elements of interference. Following the simplification proposed in [5], we define the multiplicative iterative computation of H_n and D_n:

$$\hat{X}_n = \bar{D}_n H_n; H_n \leftarrow H_n \bullet \frac{\bar{D}_n^T Z}{\bar{D}_n^T \hat{X}_n + \lambda_n}; \quad (9)$$

$$D_n \leftarrow \bar{D}_n \bullet \frac{ZH_n^T + \bar{D}_n \bullet (\mathbf{1}(\hat{X}_n H_n^T \bullet \bar{D}_n))}{\hat{X}_n H_n^T + \bar{D}_n \bullet (\mathbf{1}(ZH_n^T \bullet \bar{D}_n))}. \quad (10)$$

Where $\bar{D}_n$ is the Euclidean column-wise normalization of D_n in current iteration (see Sec. 2.4), the $\bullet$ operator indicates element-wise multiplication, the fraction line indicates element-wise division, and $\mathbf{1}$ is a square matrix of ones. The regularization parameter λ_n weights the importance of the sparsity term to the reconstruction.

The final D_n matrix represents the dictionary of the interference learned from data and it will be used by the next module to estimate the two additive sources composing the mixed signal.

2.4 Estimation of undesired source and target source

In order to estimate the sources, we use again a constrained NMF (NMF*) to compute the dictionary of the target source and the sparse code of both sources. Assuming, as usual, the additivity of sources, the dictionary of the mixed signal can be seen as the concatenation of the individual source dictionaries. Moreover, the sparse code of the mixed signal can be seen as the concatenation of the individual source sparse codes:

$$X = X_s + X_n = \begin{bmatrix} D_s\, D_n \end{bmatrix} \begin{bmatrix} H_s \\ H_n \end{bmatrix} + E = DH + E \quad (11)$$

In the previous equation, E is an unknown matrix representing approximation errors. We can not solve Eq. 11 directly with NMF, due to a permutation ambiguity. In fact, we can write

$$DH = (DP)(P^{-1}H) \quad (12)$$

where P is a generalized permutation matrix, i.e., a matrix with only one non-zero positive element in each row and each column.

Schmidt, Larsen and Hsiao [10] suggest to pre-compute D_n, as we have done in the previous section for the interference in the $Z(f,t)$ signal; then learn $D_s(f,m)$, $H_s(m,t)$ and $H_n(k,t)$, where m is the number of user defined elements of the target source, with a modified constrained NMF, which we apply to $Y(t,f)$ in Eq. 6 (i.e. the observed signal in the target-present frames). We describe here the developed one-dictionary constrained (D_n^*) algorithm:

1. Initialize $D_s(f,m)$, $H_s(m,t)$ and $H_n(k,t)$ with random values in the range $[0 \div 1]$; to multiply $H_s(m,t)$ and $H_n(k,t)$ by A to suppress target-absent frames.
2. Define Euclidean column-wise normalization of the target dictionary to prevent joint numerical drifts in H_s and D_s:

$$\bar{D}_s(f,m) = \frac{D_s(f,m)}{\sqrt{\sum_f D_s(f,m)^2}} = \frac{D_s(f,m)}{||D_s(m)||_2}. \quad (13)$$

3. Calculate the overall reconstruction according to:

$$\hat{X} = \bar{D}_s H_s + \bar{D}_n H_n. \quad (14)$$

4. Update the sparse code of target according to the rule:

$$H_s \leftarrow H_s \bullet \frac{\bar{D}_s^T Y}{\bar{D}_s^T \hat{X} + \ell_s}. \quad (15)$$

5. Calculate the overall reconstruction as in Eq. 14.

6. Update the sparse code of interference according to the rule:

$$H_n \leftarrow H_n \bullet \frac{\bar{D}_n^T Y}{\bar{D}_n^T \hat{X} + \ell_n}. \quad (16)$$

7. Calculate the overall reconstruction as in Eq. 14.

8. Update the target non-normalized dictionary according to the rule:

$$D_s \leftarrow \bar{D}_s \bullet \frac{Y H_s^T + \bar{D}_s \bullet (\mathbf{1}(\hat{X} H_s^T \bullet \bar{D}_s))}{\hat{X} H_s^T + \bar{D}_s \bullet (\mathbf{1}(Y H_s^T \bullet \bar{D}_s))}. \quad (17)$$

9. Repeat from step 2 until it reach the convergence of the Euclidean Cost function to minimize:

$$C^{(i)} = \frac{1}{2} \sum_{f,t} (Y(f,t) - \hat{X}(f,t))^2 + \ell_n \sum_{k,t} H_n(k,t) + \ell_s \sum_{m,t} H_s(m,t). \quad (18)$$

We stop the algorithm at iteration i when $|C^i - C^{i-1}| < \varepsilon C^i$. The regularization parameters ℓ_s and ℓ_n determine the degree of sparsity in the activity matrix. D_n, the dictionary of the undesired component, is left unchanged by this algorithm because it is predefined and fixed by the previous training stage; moreover, we do not seek a sparse code for the fixed dictionary, but the code that minimizes the reconstruction error, setting $\ell_n = 0$. In general λ_n, ℓ_s, k and m are depending on unknown sources. In our experimental datasets, good results were obtained for $\lambda_n = 0.2$ and $\ell_s = 0.05$, $k = 256$ and $m = 256$, confirming in a wider field of application the results of Schmidt et al. [10].

2.5 Perceptually motivated Bayesian Suppression Rules

The output of the two previous stages are the estimation of D_s, H_s, D_n and H_n; we can estimate the spectrogram of the target source and interference in target-present frames as:

$$\hat{X}_s = D_s H_s \quad (19)$$

$$\hat{X}_n = D_n H_n \quad (20)$$

Figure 2 shows the result spectrograms of $\hat{X}_s$ and $\hat{X}_n$ where the undesired component is the period stationary wide-band noise present in the observed extract 4 of Sec. 3.

We can reconstruct the target source using a noise suppression rule, a well known technique in speech enhancement and audio denoising in general. A suppression rule may be viewed as a non-negative real-valued time-frequency-varying gain $G(f,t)$, applied to the observable, target-present signal spectrum $Y(f,t)$, in order to estimate the target source spectrum:

$$\hat{S}(f,t) = G(f,t) \bullet Y(f,t) \text{ with } 0 \leq G(f,t) \leq 1 \quad (21)$$

Although in many cases, with high SNR, we can get a good reconstructed target source by means of the Wiener filter, in low SNR we get increasing target distortion and perceptually annoying musical noise (a tonal, random, isolated, time-varying noise). Generally speaking, we can reduce noise suppression in favor of better audio fidelity or speech intelligibility introducing the masking phenomenon of the human hearing model to calculate a noise masking threshold from the estimated target source. A listener tolerates additive interference, as long as its energy remains below the masking threshold defined by the target source energy, and we don't need to suppress this masked interference because it is non-audible. In this sense we suppress only the non-masked excess of interference.

A widely used, simple but effective masking model was proposed by Johnston [11] to mask the distorsion introduced in speech and audio process and adopted with success in speech enhancement. In this psychoacoustic model, a weak interference at a certain frequency is made inaudible by a stronger target occurring simultaneously (i.e., in the same frame) within the same perceptual frequency range, termed *Critical Band*, and across Critical Bands, applying a convolution with a spreading function. The Johnston's masking threshold calculation does not take into account backward or forward temporal masking. According to Wolfe and Godsill [3], we can formulate a perceptually motivated distortion measure as a generalization of the Minimum Mean Square Error (MMSE) cost function that incorporates the masking threshold:

$$C_{WG}(S,\hat{S},T) = \begin{cases} \left(\hat{S} - S - \frac{T}{2}\right)^2 - \left(\frac{T}{2}\right)^2, \\ \qquad \text{if } \left|\hat{S} - S - \frac{T}{2}\right| > \frac{T}{2}; \\ 0, \qquad \text{otherwise.} \end{cases} \quad (22)$$

where S is the true but unknown STFT amplitude of the source, $\hat{S}$ is the STFT amplitude of estimated source and T is the masking threshold; for simplicity, we omit the frequency f and the frame t indices. We can see in Eq. 22 that no cost is assigned if the estimation error is below the masking threshold and a penalty cost is assigned only when the estimation error is above the masking threshold. This prevent unwanted source attenuation when the undesired component is masked and suppress only human ear perceptible undesired component. Unfortunately, the analytical minimization of $E[C_{WG}(S,\hat{S},T)]$ is intractable and a numerical implementation was adopted by the authors [3].

In our perceptually motivated model, depicted in fig. 3, we followed a different approach based on [12], where a perceptually criterion was implicitly implemented by weighting error STFT amplitude with a filter that has the shape of the inverse STFT amplitude of the source, so that less emphasis is placed near the formant peaks (implicit masked) and more emphasis is placed on spectral valleys (implicit unmasked):

$$C_{WE}(S,\hat{S},p) = (\hat{S} - S)^2 \bullet S^p \text{with} - 2 < p \leq 0 \quad (23)$$

where p is a real value time-frequency parameter that

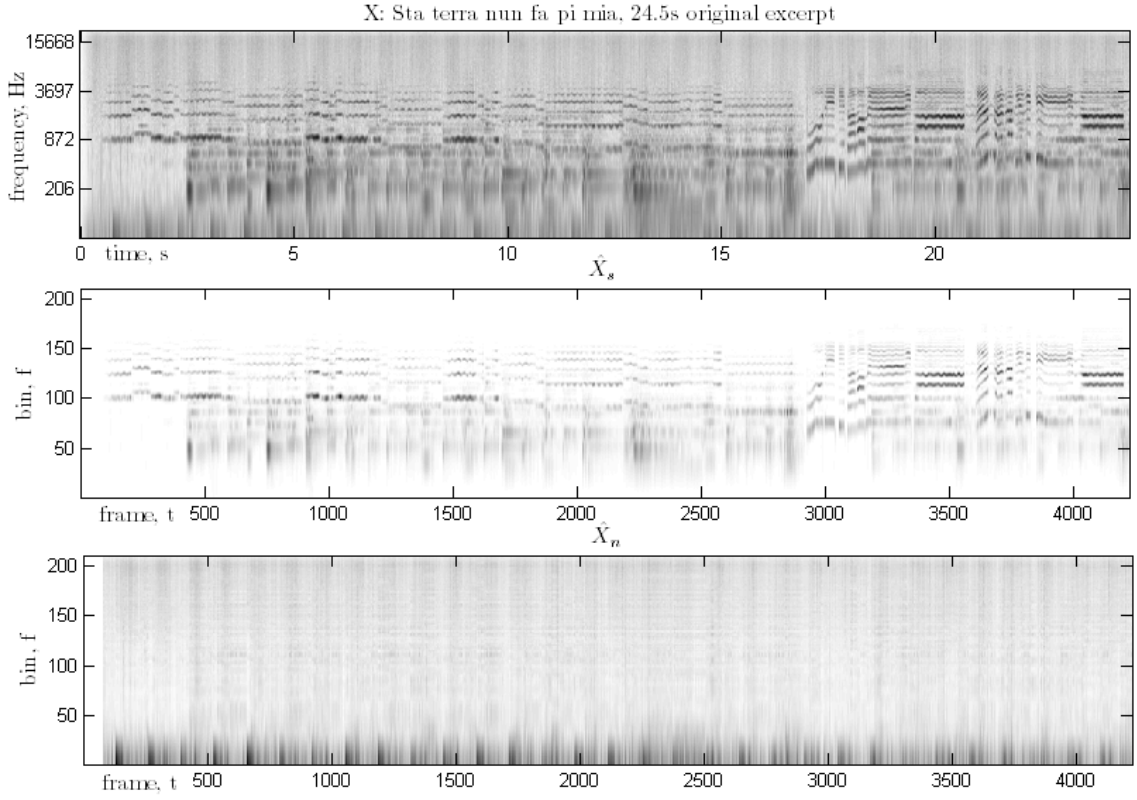


Figure 2. *Spectrograms of original noisy signal X (top), estimated target source $\hat{X}_s$ (center) and estimated interference $\hat{X}_n$ (bottom) of 24.5 seconds excerpt 'Sta terra nun fa pi mia' (see Sec. 3 item 4), spectrograms are in log-frequency representation, from $f_{min} = 50Hz$ and $24bin/octave$ resolution. Audio pattern and period stationary wide-band noise are clearly separated.*

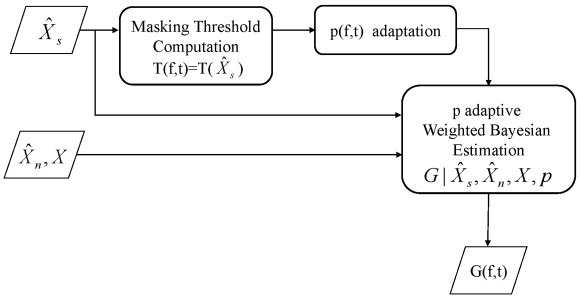


Figure 3. *The proposed perceptual suppression rule scheme based on threshold-mask-adaptive weighted Bayesian estimator.*

emphasizes spectral valleys when negative. This cost function is known as Weighted Euclidean distortion measure and the analytical minimization of $E[C_{WE}(S,\hat{S},p)]$, assuming $S(f,t)$ modelled as statistically independent zero-mean Gaussian random variables, evaluates to the time-frequency gain $G_{WE}(f,t)$:

$$G_{WE} = \frac{\sqrt{\nu}}{\gamma} \bullet \frac{\Gamma\left(\frac{p+1}{2}+1\right) \bullet \Phi\left(-\frac{p+1}{2}, 1; -\nu\right)}{\Gamma\left(\frac{p}{2}+1\right) \bullet \Phi\left(-\frac{p}{2}, 1; -\nu\right)}, \quad p > -2 \tag{24}$$

where:

$$\gamma = \frac{X}{\hat{X}_n};\ \xi = \frac{\hat{X}_s}{\hat{X}_n};\ \nu = \frac{\xi}{1+\xi} \bullet \gamma \tag{25}$$

$\Gamma(\cdot)$ denotes the gamma function and $\Phi(a,b;z)$ denotes the confluent hypergeometric function. When $p = 0$, we get the classical Ephraim and Malha MMSE STSA estimator [13].

We consider now a similar cost function, also proposed in [12], called Weighted Cosh distortion measure:

$$C_{WCOSH}(S,\hat{S},p) = \left(\frac{S}{\hat{S}} + \frac{\hat{S}}{S} - 1\right) \bullet S^p \ \text{ with } -1 < p \leq 0 \tag{26}$$

this cost function, again, emphasize spectral valleys when p is negative and evaluates to the $G_{WCOSH}(f,t)$ gain:

$$G_{WCOSH} = \frac{\sqrt{\nu}}{\gamma} \bullet \sqrt{\frac{\Gamma\left(\frac{p+3}{2}\right) \bullet \Phi\left(-\frac{p+1}{2}, 1; -\nu\right)}{\Gamma\left(\frac{p+1}{2}\right) \bullet \Phi\left(-\frac{p-1}{2}, 1; -\nu\right)}},\ p > -1 \tag{27}$$

The third Bayesian Estimator implemented in the framework is the β-Order MMSE STSA estimator (βSA), proposed in [14], and further developed in [15]. The MMSE-STSA estimator [13] [16] was generalized by the real exponent parameter β (for uniformity with previous estimators, we continue to call p):

$$C_{SA}(S,\hat{S},p) = (\hat{S}^p - S^p)^2 \text{ with } -2 < p < 0 \tag{28}$$

taking $p < 0$, the behavior to penalize the cost function is similar to Weighted Euclidean Distortion Measure of

Eq. 23, and both perform an accurate estimation of source in spectral valleys. The gain function $G_{SA}(f,t)$ derived from the βSA estimator is expressible as:

$$G_{SA} = \frac{\sqrt{\nu}}{\gamma} \bullet \left[\Gamma\left(\frac{p}{2}+1\right) \bullet \Phi\left(-\frac{p}{2}, 1; -\nu\right)\right]^{1/p}, \quad p > -2 \tag{29}$$

When $p \to 0$, the βSA is equivalent to the Ephraim and Malah MMSE log-STSA [16].

Extensive tests of the three perceptually motivated Bayesian estimators, with PESQ measure and informal audio assessment on speech phrases at low SNR (i.e. < 10dB), show that all of them perform equally well, in the sense that they don't introduce musical noise and permit to control the trade-off between undesired component suppression and source attenuation by varying the parameter p. A performance evaluation and comparative audio samples are available in `http://dialogo.fisica.uniud.it/BASS/ComparisionWithGustafsson02`.

Indeed, the optimal choice of the real parameter $p(f,t)$ is an important performance issue. Therefore, we considered to express explicitly the relation with the masking threshold $T(f,t)$, although in an heuristic manner. We have seen that the undesired component can be reduced by decreasing p, however this leads to more source distortion. Therefore, the adaptation is based on the following consideration: if the masking threshold is high, interference will be masked and consequently inaudible. Consequently, there is no need to reduce, which helps to keep distortion as low as possible. In this case the parameter p is kept to his maximal value: $p = p_{max}$. However, if the masking threshold is low, undesired component will be unpleasant or even annoying to the ear and it is necessary to reduce it. This is done by a decrease of p toward his minimum value: $p = p_{min}$. For each frame t, the minimum of the masking threshold $T(f,t)$ corresponds to the minimum of the power parameter $p(f,t)$. In order to avoid discontinuities in the gain function G due to this adaptation, a smoothing operation is applied, controlled by user value x. The adaptation of the parameter $p(f,t)$ is performed with the following relation:

$$p(f,t) = \left(\frac{T(f,t) - T_{min}(t)}{T_{max}(t) - T_{min}(t)}\right)^{x} (p_{max} - p_{min}) + p_{min} \tag{30}$$

where $T_{max}(t)$ and $T_{min}(t)$ are the maximal and minimal values of noise masking threshold $T(f,t)$ at current frame t. In this way, $p(f,t)$ adapts to a minimal interference reduction for the maximal values of the masking threshold (i.e. in correspondence of source formant peaks) and a maximal reduction for the minimal values of the threshold (i.e. in correspondence of spectral valleys). Figure 4 show the simple smoothing curves obtained with Eq. 30 varying the smoothing parameter x.

The minimal and maximal values of p and x determine the tradeoff between residual noise and source distortion. A number of experiments with different noise types and levels have been performed to select the appropriate values for these parameters.

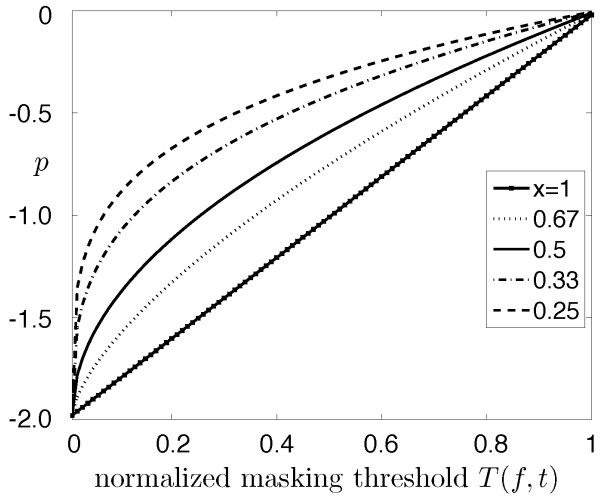


Figure 4. *Parameter p versus the normalized masking threshold T for smoothing parameter $x = (0.25, 0.33, 0.5, 0.67, 1)$, $p_{min} = -1.98$, $p_{max} = 0$.*

For additive interference at SNR proximal to 0 dB, the following values have been chosen in order to obtain a good tradeoff for a human listener for each Bayesian estimator:

1. For Weighted Euclidean: $p_{WEmax} = 0$, $p_{WEmin} = -1.98$;
2. For Weighted Cosh: $p_{WCOSHmax} = 0$, $p_{WCOSHmin} = -0.99$;
3. For βSA: $p_{max} = -0.001$, $p_{min} = -1.98$;

The smoothing exponent is kept fixed for all estimators: $x = 0.5$.

This tradeoff can be easily changed depending on the application; in general, only a regulation of p_{min} is needed to optimal tradeoff.

The use of Johnston' simultaneous masking threshold estimation allows the construction of effective and sophisticated perceptual noise suppression rules. However, if the threshold is not correctly estimated, performance greatly suffers in terms of very annoying musical noise injected in the target source waveform, compromising any noise suppression rule. To properly estimate the threshold, we need an extremely accurate estimation of the target source spectrum $\hat{X}_s(f,t)$ that we obtained with NMF.

3. SUBJECTIVE EVALUATION

To validate the system, a listening test was conducted. As audio material, several sound documents of ethnic music were considered.

Material. Four music pieces recorded in Shellac disc were used. In order to minimize fatigue and maximize attention by the participating subjects, we selected the 20 first seconds of each stimulus. Since the task was more a comparison than an individual analysis, those short extracts seemed to be sufficient.

1. *Chi campa deritto campo aflitto* (Who lives honestly lives poorly, by Perrocato and Canoro), Eduardo Migliaccio (voc) - 78 rpm 10" Victor 14-81712-B (BVE 46692-2), rec. in New York, August, 14, 1928, length 3'36". In the excerpt considered: singing voice and music.

2. *Il funerale di Rodolfo Valentino* (The funeral of Rodolfo Valentino), Compagnia Columbia (2 male singers, 2 female singers, bells and Orchestra) - 78 rpm 10" Columbia 14230-F (w 107117 2), rec. in New York, September, 1926, length 2'55". In the excerpt considered: speech voices.

3. *La signorina sfinciusa* (The funny girl), Leonardo Dia (voc), Alfredo Cibelli (mandolin), unknowns (2 guitars) - 78 rpm 10" Victor V-12067-A (BVE 53944-2), rec. in New York, July, 24, 1929, length 3'20". In the excerpt considered: singing voice and music.

4. *Sta terra nun fa pi mia* (This land is not for me, by R. Gioiosa, arr. R. Romani), Rosina Gioiosa Trubia (voc), Alfredo Cibelli (mandolin), unknowns (2 guitars) - 78 rpm 10" Brunswick 58073B (E 26621/2), rec. in New York, February, 23, 1928, length 3'22". In the excerpt considered: singing voice and music.

Noisy stimuli was pre-processed with the Extended Kalman Filter, detailed in [17] (in de-click mode), then broad band restoration was performed using our framework with the suppression rule detailed in 2.5, as well as the following three commercial products, selected among the most appreciated products in the audio archives and post-processing studios:

1. X-Noise of Waves Restoration bundle (Waves V6 Update 2);

2. Denoiser (with the *Musical noise suppression* filter enabled) of iZotope RX v1.06;

3. Auto Dehiss of CEDAR Tools;

The CEDAR Tools plug-ins are used in a Pro Tools HD system. The parameters used to control the different systems were subjectively set to obtain the best tradeoff between noise removal and music signal preservation. In this way 16 restored stimuli were produced.

Test method. The tests were conducted using the EBU MUSHRA test method [18], which is a recommended evaluation method adopted by ITU [19]. This protocol is based on the "double-blind triple-stimulus with hidden reference" method, which is stable and permits accurate detection of small impairments. An important feature of this method is the inclusion of the hidden reference and of two bandwidth-limited anchors signals (7 kHz and 3.5 kHz).

The noisy stimuli under test are all real-world signals. This implies that we can not compare test enhanced sound with a high quality reference sound (graded 5.0 at the top of the grading scale), but with the noisy reference sound (graded 0.0). Moreover, negative scores are allowed to evaluate test sounds that rate worse then the noisy reference. At least the hidden reference must be graded 0.0 by the evaluator. All the other test stimuli and hidden anchors can be evaluated subjectively to rate the overall quality of sound excerpts.

Training phase. The purpose of the training phase, according to the MUSHRA specification, was to allow each listener: i) to become familiar with all the sound excerpts under test and their quality-level ranges; ii) to learn how to use the test equipment and the grading scale.

Listeners. Two subject groups were selected:

1. Musically trained (MT): 12 researchers (musicologists and/or musicians) of the University of Padova and 12 technicians of different international audio archives.

2. Musically untrained (MU): 16 students in Information Engineering (University of Padova).

Equipment. The audio signals were recorded at 44.1 kHz/24 bit (uncompressed sound files) and played through Apple PowerBook Pro 2.4 GHz Intel Core 2 Duo with 2 GB 1067 MHz DDR3 equipped with a D/A converter RME Fireface 400, and headphones AKG K 501. The listeners could play in any order all the stimuli under test, including the hidden reference and the two bandwidth-limited anchor signals.

Test duration. The training session for each listener took approximately 40', including an explanation about the tests and equipment, and a practice grading session. The grading phase consisted of 4 test sessions (one for each music piece), each one containing 9 test signals (1 noisy signal, 6 restored signals, 2 anchors). Each session took, on average, about 8 minutes. Subjects were allowed a rest period between each session, but not during a session.

Main results. The statistical analysis method described in the MUSHRA specification was used to process the test data. The results are presented in Tab. 1 as mean grades. The results from six listeners (five of them belong to MU group, one to MT) were rejected because the mean of their rates (in absolute value) on hidden references is greater than $+/-0.5$.

The quality range between the best and worst restoration system is only 0.80 (MT group) and $0.40(MUgroup)$. In general there are only two systems with a score > 3.5: our Tool and CEDAR. Our algorithm produces scores similar to CEDAR in both test sessions (better for MU group) and better than the others softwares.

Table 1. *Mean for restored stimuli and anchors, 34 subjects. MT = Musically trained; MU = Musically untrained.*

Restoration system	**MT group**	**MU group**	**Average**
Our Tool	$+3.00$	$+4.20$	$+3.60$
CEDAR Tools	$+3.40$	$+4.00$	$+3.70$
Waves	$+2.80$	$+3.80$	$+3.30$
iZotope RX	$+2.20$	$+3.80$	$+3.00$
Anchor 7 kHz	-2.69	$+0.20$	-1.02
Anchor 3.5 kHz	-5.00	-4.20	-4.60

4. CONCLUSIONS

This study is focused on the restoration of single channel audio recordings of ethnic music: for this purpose, we applied EKF framework to audio signal enhancement problems. In this paper we investigate the use of the Non-negative Matrix Factorization (NMF): we show that NMF is a suitable technique to extract the clean audio signal from undesired non stationary noise in a monaural recording with low SNR. More specifically, we introduce a perceptual suppression rule based on an advanced psychoacoustic models (Sec. 2.5. To evaluate the proposed approach a subjective audio enhancement experiments was carried out (see Section 3). The results of this experiments show that the proposed method results in improved audio quality and that it is a useful alternative to the classical STSA methods.

Future work will carry out an intensive application of this audio restoration environment on two real archives of ethnic music phonographic discs.

5. REFERENCES

[1] W. Storm, "The establishment of international re-recording standards," *Phonographic Bulletin*, vol. 27, pp. 5–12, 1980.

[2] S. Canazza and A. Vidolin, "Preserving electroacoustic music," *Journal of New Music Research*, vol. 30, no. 4, pp. 351–363, 2001.

[3] P. J. Wolfe and S. J. Godsill, "Towards a perceptually optimal spectral amplitude estimator for audio signal enhancement," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. II, (Istanbul, Turkey), pp. 821–824, 2000. ISBN 0-7803-6296-9.

[4] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, no. 401, pp. 788–791, 1999.

[5] J. Eggert and E. Krner, "Sparse coding and nmf," in *IEEE International Conference on Neural Networks*, pp. 2529–2533, IEEE, 2004.

[6] R. Meddis, L. P. O'Mard, and E. A. Lopez Poveda, "A computational algorithm for computing nonlinear auditory frequency selectivity," *Journal of the Acoustical Society of America*, vol. 109, no. 6, pp. 2852–2861, 2001.

[7] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, vol. 87, pp. 1738–1752, Apr 1990.

[8] E. Plourde and B. Champagne, "Integrating the cochleas compressive nonlinearity in the bayesian approach for speech enhancement," in *15th EUSIPCO, Poznan, Poland*, pp. 70–74, 2007.

[9] J. Sohn, N. S. Kim, and W. Sung, "A statistical model-based voice activity detection," *IEEE Signal Processing Letters*, vol. 6, pp. 1–3, 1999.

[10] M. N. Schmidt, J. Larsen, and F. T. Hsiao, "Wind noise reduction using non-negative sparse coding," in *IEEE Workshop on Machine Learning for Signal Processing*, pp. 431–436, Aug 2007.

[11] J. D. Johnston, "Transform coding of audio signals using perceptual noise criteria," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 2, pp. 314–323, 1988.

[12] P. C. Loizou, "Speech enhancement based on perceptually motivated bayesian estimators of the magnitude spectrum," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5-2, pp. 857–869, 2005.

[13] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 6, pp. 1109–1121, 1984.

[14] S. H. K. C. H. You and S. Rahardja, "β-order mmse spectral amplitude estimation for speech enhancement," *IEEE Trans. Speech Audio Processing*, vol. 13, no. 4, pp. 475–486, 2005.

[15] E. Plourde and B. Champagne, "Further analysis of the β-order mmse stsa estimator for speech enhancement," in *Canadian Conference on Electrical and Computer Engineering. CCECE 2007*, pp. 1594–1597, 2007.

[16] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 2, pp. 443–445, 1985.

[17] S. Canazza, G. De Poli, and G. Mian, "Restoration of audio documents by means of extended kalman filter," *IEEE Transactions on Audio, Speech and Language Processing*, vol. in press, 2010.

[18] ITU-R, "Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems," *Recommendation BS.1116-1*, 2000.

[19] EBU Project Group B/AIM, "EBU report on the subjective listening tests of some commercial internet audio codecs," October 2000.

CONSTANT-Q TRANSFORM TOOLBOX FOR MUSIC PROCESSING

Christian Schörkhuber
Institute of Electronic Music and Acoustics
University of Music and Performing Arts, Graz
cschoerkhuber@student.tugraz.at

Anssi Klapuri[1]
Centre for Digital Music
Queen Mary University of London
anssi.klapuri@elec.qmul.ac.uk

ABSTRACT

This paper proposes a computationally efficient method for computing the constant-Q transform (CQT) of a time-domain signal. CQT refers to a time-frequency representation where the frequency bins are geometrically spaced and the Q-factors (ratios of the center frequencies to bandwidths) of all bins are equal. An inverse transform is proposed which enables a reasonable-quality (around 55dB signal-to-noise ratio) reconstruction of the original signal from its CQT coefficients. Here CQTs with high Q-factors, equivalent to 12–96 bins per octave, are of particular interest. The proposed method is flexible with regard to the number of bins per octave, the applied window function, and the Q-factor, and is particularly suitable for the analysis of music signals. A reference implementation of the proposed methods is published as a Matlab toolbox. The toolbox includes user-interface tools that facilitate spectral data visualization and the indexing and working with the data structure produced by the CQT.

1. INTRODUCTION

Constant-Q transform (CQT) here refers to a technique that transforms a time-domain signal $x(n)$ into the time-frequency domain so that the center frequencies of the frequency bins are geometrically spaced and their Q-factors are all equal. In effect, this means that the frequency resolution is better for low frequencies and the time resolution is better for high frequencies. The CQT is essentially a wavelet transform, but here the term CQT is preferred since it underlines the fact that we are considering transforms with relatively high Q-factors, equivalent to 12–96 bins per octave. This renders many of the conventional wavelet transform techniques inadequate; for example methods based on iterated filterbanks would require filtering the input signal hundreds of times.

The CQT is well-motivated from both musical and perceptual viewpoints. The fundamental frequencies (F0s) of the tones in Western music are geometrically spaced: in the standard 12-tone equal temperament, for example, the F0s obey $F_k = 440\text{Hz} \times 2^{k/12}$, where $k \in [-50, 40]$ is an integer. From auditory perspective, the frequency resolution of the peripheral hearing system of humans is approximately constant-Q over a wide range from 20kHz down to approximately 500Hz, below which the Q-values get progressively smaller [1]. From perceptual audio coding, we know that the shortest transform window lengths have to be of the order 3ms in order to retain high quality, whereas higher frequency resolution is required to carry out coding at low frequencies [2]. All this is in sharp contrast with the conventional discrete Fourier transform (DFT) which has linearly spaced frequency bins and therefore cannot satisfy the varying time and frequency resolution requirements over the wide range of audible frequencies.

There are at least three reasons why the CQT has not widely replaced the DFT in audio signal processing. Firstly, it is computationally more intensive than the DFT. Secondly, the CQT lacks an inverse transform that would allow perfect reconstruction of the original signal from its transform coefficients. Thirdly, CQT produces a data structure that is more difficult to work with than the time-frequency matrix (spectrogram) obtained by using short-time Fourier transform in successive time frames. The last problem is due to the fact that in CQT, the time resolution varies for different frequency bins, in effect meaning that the "sampling" of different frequency bins is not synchronized. In this paper, we propose solutions to these three problems.

As already mentioned above, constant-Q transform can be viewed as a wavelet transform. The wavelet literature is well-matured (see e.g. [3]) and constant-Q (wavelet) transforms have been proposed that lead to perfect reconstruction. However most of the work has focused on critically-sampled dyadic wavelet transforms, where the frequency resolution is only one bin per octave – this is clearly insufficient for music signal analysis. Recently, perfect reconstruction wavelet transforms have been proposed that have rational dilation factors, meaning that the center frequencies of the bins are spaced by p/q, where p and q are integers [4, 5]. However, these are based on iterated filter banks and are therefore less attractive computationally when high Q-factors, such as 12–96 bins per octave, are required. Another interesting direction of research has been the application of frequency warping on a time-domain signal in such a way that the DFT of the warped signal is related to the DFT of the original signal via a frequency warping function [6, 7]. A problem with these is that the warping filters have infinite impulse responses which makes it hard to design an inverse transform.

Brown and Puckette proposed a computationally effi-

[1] Equally contributing authors.

cient technique for computing constant-Q transforms with high Q factors based on the fast Fourier transform (FFT) and a frequency-domain kernel [8, 9]. A drawback of this CQT implementation is that there is no inverse transform for it. Recently, FitzGerald has shown that a good quality approximate inverse transform can be obtained if the signal to be inverted has a sparse representation in the discrete Fourier transform domain [10]. However, this is not true for music signals in general.

This paper proposes specific solutions to the three problems of CQT mentioned above. Our solution to the computational efficiency problem is based on the technique proposed by Brown and Puckette in [9] which we extend to further improve its computational efficiency. Secondly, we propose to structure the transform kernel in such a way that reasonable-quality inverse transform (approximately 55dB signal-to-noise ratio) is obtained using the conjugate transpose of the CQT transform kernel. The reconstruction is achieved introducing only a moderate amount of redundancy (by factor four or five) to the transform (here redundancy refers to the number of elements in the transform compared to the samples in the original time-domain signal). Thirdly, we propose interface tools for the data structure that facilitate working with the signal in the transform domain. A reference implementation of the proposed methods is provided as a Matlab toolbox at `http://www.elec.qmul.ac.uk/people/anssik/cqt/`.

2. SIGNAL MODEL

The CQT transform $X^{\mathsf{CQ}}(k,n)$ of a discrete time-domain signal $x(n)$ is defined by

$$X^{\mathsf{CQ}}(k,n) = \sum_{j=n-\lfloor N_k/2 \rfloor}^{n+\lfloor N_k/2 \rfloor} x(j) a_k^*(j-n+N_k/2) \quad (1)$$

where $k = 1, 2, \ldots, K$ indexes the frequency bins of the CQT, $\lfloor \cdot \rfloor$ denotes rounding towards negative infinity and $a_k^*(n)$ denotes the complex conjugate of $a_k(n)$. The basis functions $a_k(n)$ are complex-valued waveforms, here also called time-frequency *atoms*, and are defined by

$$a_k(n) = \frac{1}{N_k} w\left(\frac{n}{N_k}\right) \exp\left[-i2\pi n \frac{f_k}{f_\mathrm{s}}\right] \quad (2)$$

where f_k is the center frequency of bin k, f_s denotes the sampling rate, and $w(t)$, is a continuous window function (for example Hann or Blackman window), sampled at points determined by t. The window function is zero outside the range $t \in [0, 1]$.

The window lenghts $N_k \in \mathbb{R}$ in (1)–(2) are real-valued and inversely proportional to f_k in order to have the same Q-factor for all bins k. Note that in (1) the windows are centered at the sample n of the input signal. Different window functions will be discussed in Sec. 5.

In the CQT considered here, the center frequencies f_k obey

$$f_k = f_1 2^{\frac{k-1}{B}} \quad (3)$$

where f_1 is the center frequency of the lowest-frequency bin, and B determines the number of bins per octave. In practice, B is the most important parameter of choice when using the CQT, because it determines the time-frequency resolution trade-off of the CQT.

The Q-factor of bin k is given by

$$Q_k \stackrel{\text{def.}}{=} \frac{f_k}{\Delta f_k} = \frac{N_k f_k}{\Delta\omega f_\mathrm{s}} \quad (4)$$

where Δf_k denotes the -3dB bandwidth of the frequency response of the atom $a_k(n)$ and $\Delta\omega$ is the -3dB bandwidth of the mainlobe of the spectrum of the window function $w(t)$, being $\Delta\omega \approx 1.50$ [DFT bins] for the Hann window and $\Delta\omega \approx 1.73$ for Blackman, for example. The Q-factors Q_k are by definition the same for all bins, therefore we omit the subscript and write simply Q below.

It is typically desirable to make Q as large as possible, so as to make the bandwidth Δf_k of each bin as small as possible and thus introduce minimal frequency smearing. However, we cannot employ arbitrarily high Q factors – otherwise portions of the spectrum between the bins would not be analyzed. The value of Q that introduces minimal frequency smearing but still allows signal reconstruction is

$$Q = \frac{q}{\Delta\omega(2^{\frac{1}{B}} - 1)} \quad (5)$$

where $0 < q \lessapprox 1$ is a scaling factor, and typically $q = 1$. Values of q smaller than 1 can be used to improve the time resolution at the cost of degrading the frequency resolution. Important to note is that setting for example $q = 0.5$ and $B = 48$ leads to exactly the same time-frequency resolution trade-off as setting $q = 1$ and $B = 24$, but the former contains twice more frequency bins per octave. In this sense, values $q < 1$ can be seen to implement *oversampling* of the frequency axis, analogously to the use of zero padding when calculating the DFT. For example $q = 0.5$ corresponds to oversampling factor of 2: the effective frequency resolution is equivalent to $B/2$ bins per octave, although B bins per octave are computed.

Substituting (5) in (4) and solving for N_k, we get

$$N_k = \frac{q f_\mathrm{s}}{f_k(2^{\frac{1}{B}} - 1)} \quad (6)$$

where we see that the dependency on $\Delta\omega$ has disappeared.

It is not computationally reasonable to calculate the coefficients $X^{\mathsf{CQ}}(k,n)$ at all positions n of the input signal. To enable signal reconstruction from the CQT coefficients, successive atoms can be placed H_k samples apart ("hop size"). In order to analyze all parts of the signal properly and to achieve reasonable signal reconstruction, values $0 < H_k \lessapprox \frac{1}{2} N_k$ are useful.

3. ALGORITHM FOR COMPUTING THE TRANSFORM

The computationally-efficient forward CQT transform proposed here is based on the principles proposed by Brown and Puckette in [9]. Therefore we first explain the technique proposed in [9] and then describe the extensions in Subsection 3.2.

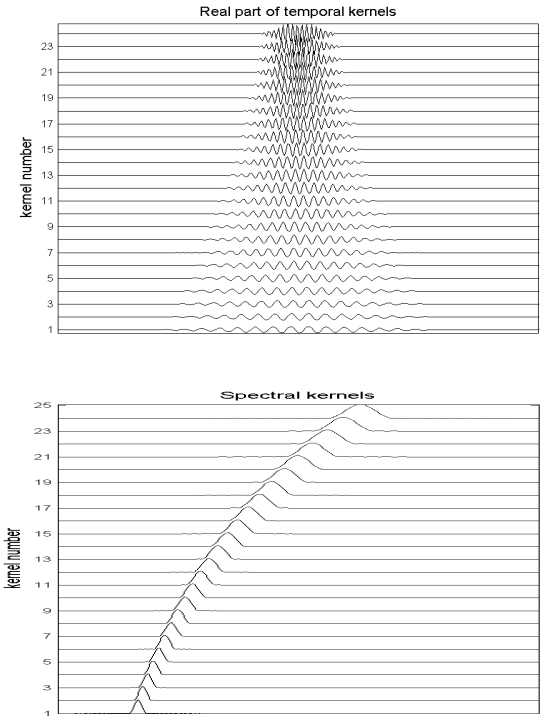


Figure 1. The upper panel illustrates the real part of the transform bases (temporal kernel) that can be used to calculate the CQT over two octaves, with 12 bins per octave. The lower panel shows the absolute values of the corresponding spectral kernel.

3.1 Algorithm of Brown and Puckette

Let us assume that we want to calculate the CQT transform coefficients $X^{\mathsf{CQ}}(k,n)$ as defined by (1) at one point n of an input signal $x(n)$. A direct implementation of (1) obviously requires calculating inner products of the input signal with each of the transform bases. The upper panel of Fig. 1 illustrates the real part of the transform bases $a_k(n)$, assuming here for simplicity only $B = 12$ bins per octave and a frequency range of two octaves.

A computationally more efficient implementation is obtained by utilizing the identity

$$\sum_{n=0}^{N-1} x(n)a^*(n) = \sum_{j=0}^{N-1} X(j)A^*(j) \tag{7}$$

where $X(j)$ denotes the discrete Fourier transform (DFT) of $x(n)$ and $A(j)$ denotes the DFT of $a(n)$. Equation (7) holds for any discrete signals $x(n)$ and $a(n)$ and stems from Parseval's theorem [3].

Using (7), the CQT transform in (1) can be written as

$$X^{\mathsf{CQ}}(k, N/2) = \sum_{j=0}^{N} X(j)A_k^*(j) \tag{8}$$

where $A_k(j)$ is the complex-valued N-point DFT of the transform basis $a_k(n)$ so that the bases $a_k(n)$ are centered at the point $N/2$ within the transform frame. Following the terminology of [9], we will refer to $A_k(j)$ as the spectral kernels and to $a_k(n)$ as the temporal kernels. The lower panel of Fig. 1 illustrates the absolute values of the spectral kernels $A_k(j)$ corresponding to temporal kernels $a_k(n)$ in the upper panel.

As observed by Brown and Puckette, the spectral kernels $A_k(j)$ are sparse: most of the values being near zero because they are Fourier transforms of modulated sinusoids. Therefore the summation in (8) can be limited to values near the peak in the spectral kernel to achieve sufficient numerical accuracy – omitting near-zero values in $A_k(j)$. This is the main idea of the efficient CQT transform proposed in [9]. It is also easy to see that the summing has to be carried out for positive frequencies only, followed by multiplication by two.

For convenience, we store the spectral kernels $A_k(j)$ as columns in matrix $\mathbf{A}$. The transform in (8) can then be written in matrix form as

$$\mathbf{X}^{\mathsf{CQ}} = \mathbf{A}^*\mathbf{X} \tag{9}$$

where $\mathbf{A}^*$ denotes the conjugate transpose of $\mathbf{A}$. Matrices $\mathbf{X}$ and $\mathbf{X}^{\mathsf{CQ}}$ have only one column each, containing the DFT values $X(j)$ and the corresponding CQT coefficients, respectively.

3.2 Processing One Octave at a Time

There are two remaining problems with the method outlined in the previous subsection. Firstly, when a wide range of frequencies is considered (for example, eight octaves from 60Hz to 16kHz), quite long DFT transform blocks are required and the spectral kernel is no longer very sparse, since the frequency responses of higher frequency bins are wider as can be seen from Fig. 1. Secondly, in order to analyze all parts of the input signal adequately, the CQT transform for the highest frequency bins has to be calculated at least every $N_K/2$ samples apart, where N_K is the window length for the highest CQT bin. Both of these factors reduce the computational efficiency of the method.

We propose two extensions to address the above problems. The first is processing by octaves. [2] We use a spectral kernel matrix $\mathbf{A}$ which produces the CQT for the highest octave only. After computing the highest-octave CQT bins over the entire signal, the input signal is lowpass filtered and downsampled by factor two, and then the same process is repeated to calculate the CQT bins for the next octave, using exactly the same DFT block size and spectral kernel (see (8)). This is repeated iteratively until the desired number of octaves has been covered. Figure 2 illustrates this process.

Since the spectral kernel $\mathbf{A}$ now represents frequency bins that are at maximum one octave apart, the length of the DFT block can be made quite short (according to N_k of the lowest CQT bin) and the matrix $\mathbf{A}$ is very sparse even for the highest-frequency bins.

Another computational efficiency improvement is obtained by using several temporally translated versions of the transform bases $a_k(n)$ within the same spectral kernel

[2] We want to credit J. Brown for mentioning this possibility already in [8], although octave-by-octave processing was not implemented in [8, 9].

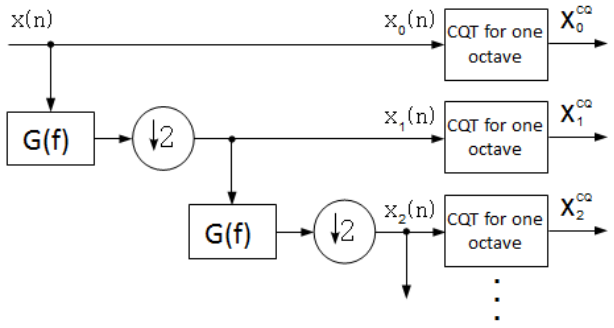


Figure 2. An overview of computing the CQT one octave at the time. Here $G(f)$ is a lowpass filter and $\downarrow 2$ denotes downsampling by factor two.

matrix $\mathbf{A}$. In other words, successive columns of $\mathbf{A}$ contain the DFTs of $a_k(n)$ that have been temporally shifted to different locations. As a result, DFT transforms of the input signal $x(n)$ to obtain the DFT spectrum $X(j)$ in (8) need to be computed less often: if there are P successive atoms within the same spectral kernel, the DFTs need to be computed P times less often.

Figure 3 illustrates the general structure of the kernel matrix applied in this paper. In the shown example, the number of bins per octave $B = 12$. By looking closely, it can be seen that the highest four kernel functions have the same center frequency, but correspond to four different temporal locations. Similarly, the two lowest kernel functions correspond to the same frequency, but different temporal locations. The detailed structure of the spectral kernel will be discussed in Sec. 5; here it suffices to say that the kernel structure is crucial for high-quality reconstruction (inverse CQT) of the input signal $x(n)$ from the CQT coefficients.

The transform for a single octave (indicated by "CQT for one octave" in Fig. 2) is defined as follows. Let $x_d(n)$ denote a signal that is obtained by decimating the input signal d times by factor two. The sampling rate of $x_d(n)$ is therefore $f_s/2^d$. The signal $x_d(n)$ is blocked into DFT transform frames of length N_{DFT} which are positioned H_{DFT} samples apart (i.e., successive frames overlap by $N_{\mathsf{DFT}} - H_{\mathsf{DFT}}$ samples). Each frame is Fourier transformed using a rectangular window and the resulting spectrogram is stored in a matrix $\mathbf{X}$, where column m contains the complex-valued spectrum of frame m (positive frequencies only). Then the CQT transform $\mathbf{X}_d^{\mathsf{CQ}}$ for this octave d is calculated as

$$\mathbf{X}_d^{\mathsf{CQ}} = \mathbf{A}^* \mathbf{X}_d \tag{10}$$

where $\mathbf{A}^*$ is the conjugate transpose of the complex-valued spectral kernel matrix for one octave as described above. The column m of $\mathbf{X}_d^{\mathsf{CQ}}$ contains the CQT coefficients representing DFT block m and the different rows of $\mathbf{X}_d^{\mathsf{CQ}}$ correspond to the different spectral kernels that are stored in the different columns of matrix $\mathbf{A}$.

The above process is repeated for each successive octave, as illustrated in Fig. 2. Note that the kernel remains the same for all octaves. Also, the DFT length N_{DFT} (in samples) remains the same despite the decimations, therefore the effective FFT length (in seconds) doubles in each decimation. The first octave is computed using signal $x_0(n)$, which is identical to the input, $x(n)$.

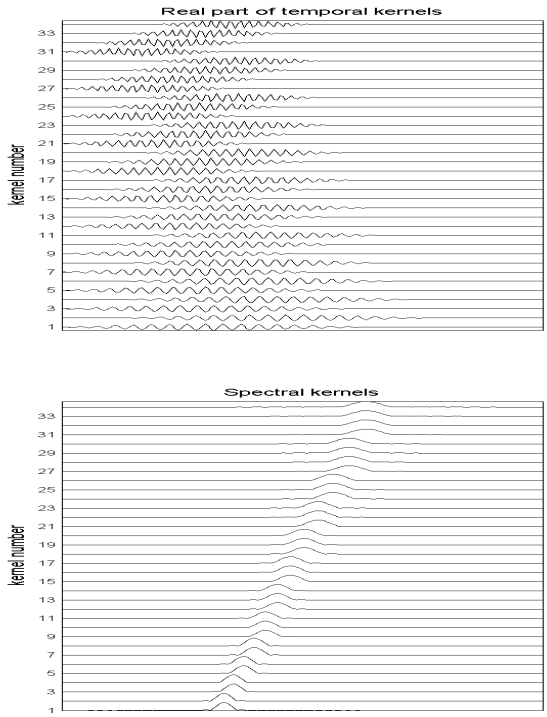


Figure 3. Illustration of the general structure of the kernel matrices used in this paper. The upper panel shows the real part of the temporal kernel used to compute the CQT for one octave. The lower panel shows the absolute values of the corresponding spectral kernel.

The decimated signals $x_d(n)$ are obtained from $x_{d-1}(n)$ by lowpass filtering and downsampling by factor two. For the lowpass filter $G(f)$, we use zero-phase forward-and-reverse filtering with a sixth-order Butterworth IIR filter that has a cut-off frequency $f_s/4$. Forward-and-reverse filtering means that after filtering in the forward direction, the filtered sequence is reversed and run back through the filter and the result of the second filtering is then reversed once more. The result has precisely zero phase distortion and magnitude modified by the square of the filter's magnitude response. Figure 4 shows the magnitude response of the lowpass filter $G(f)$ (square of the magnitude response of sixth-order Butterworth filter). Downsampling by factor two is then done simply by removing every second sample of the time-domain signal.

A final practical consideration is to deal with the beginning and end of the input signal $x(n)$. We address this problem by padding $2^{D-1}N_1$ zeros at be beginning of the signal and $2^{D-1}N_{\mathsf{DFT}}$ zeros at the end of the signal, where N_1 is the window length of the lowest-frequency bin within the one-octave kernel, D is the number of octaves calculated, and N_{DFT} is the length of the DFT frame. The zero padding is done before any of the CQT computations, and the zeros are then removed at the inverse transform stage. Note that a smaller number of zeros is needed, if the

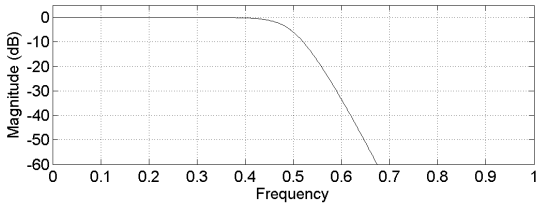


Figure 4. Magnitude response of the lowpass filter $G(f)$.

zero padding is done separately for each octave, but this would make the implementation less clear and therefore we assume that the number of zeros padded is negligible in comparison to the length of the input signal $x(n)$.

3.3 Computational Complexity

Let L denote the length of the input signal $x(n)$ after the zero padding at the beginning and the end. The number of DFT frames m to cover the entire signal before any decimation is $\lfloor (L - N_{\mathsf{DFT}})/H_{\mathsf{DFT}} \rfloor + 1$. For the next octave, the number of fast Fourier transforms (FFTs) is roughly twice smaller, $\lfloor (L/2 - N_{\mathsf{DFT}})/H_{\mathsf{DFT}} \rfloor + 1$, in fact a bit more than twice smaller. For the next octave, the number of DFT transforms is rought four times smaller, and so forth. Since $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots \approx 2$, the total number C of FFTs to compute is

$$C \leq 2\,(\lfloor (L - N_{\mathsf{DFT}})/H_{\mathsf{DFT}} \rfloor + 1) \qquad (11)$$

regardless of the number of octaves computed.

For each of the C DFT frames, the complex-valued DFT spectrum (a column vector) has to be multiplied by the conjugate transpose of the spectral kernel $\mathbf{A}^*$ (see (10)). However, since $\mathbf{A}$ is sparse, the number of multiplications is quite small. In our reference Matlab implementation, the matrix is implemented as a sparse matrix, therefore also the memory complexity of storing $\mathbf{A}$ is quite low. The exact number of non-zero elements in $\mathbf{A}$ depends on the kernel structure and the threshold below which the near-zero elements are rounded to zero (see 8). The number of lowpass filterings is proportional to the number of octaves D and causes a non-negligible computational load too.

To compare the complexity of the proposed method with that of the original method by Brown and Puckette [9], consider a case where the CQT is computed over an eight-octave range. If atoms over all octaves are stored into a single kernel, the frequency kernels in the highest octave will have $2^7 = 128$ times more non-zero elements than the corresponding atoms in the lowest octave, and the number of multiplications in (9) increases in the same proportion. The lengths of the DFT transform frames, in turn, have to be 128 times larger in order to accommodate the lowest-frequency atoms without decimation.

4. INVERSE CQT TRANSFORM

Figure 5 shows an overview of the inverse CQT transform (ICQT), where an approximation $\hat{x}(n)$ of the input signal $x(n)$ is reconstructed from the octave-wise CQT coefficient matrices $\mathbf{X}_d^{\mathsf{CQ}}$. The process is analogous to the forward transform, except that all is done in reverse order.

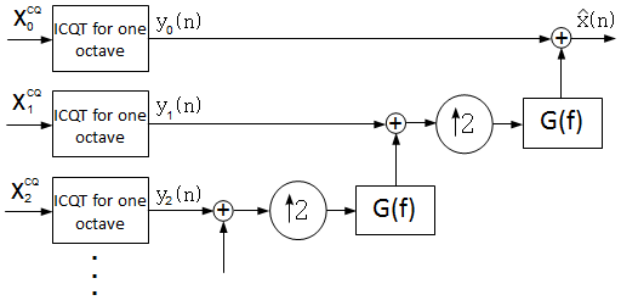


Figure 5. An overview of the inverse CQT transform (ICQT), where an approximation $\hat{x}(n)$ of the input signal $x(n)$ is reconstructed from the octave-wise CQT coefficient matrices $\mathbf{X}_d^{\mathsf{CQ}}$.

The block indicated by "ICQT for one octave" in Fig. 5 corresponds to the reconstruction of a time-domain signal $y_d(n)$ that represents only one-octave range of the input signal $x(n)$. The signal $y_d(n)$ is obtained as follows. First, an inverse spectral kernel $\mathbf{V}$ is applied to reconstruct the complex-valued DFT bins within this single octave:

$$\mathbf{Y}_d = \mathbf{V}^* \mathbf{X}_d^{\mathsf{CQ}} \qquad (12)$$

where the column m of $\mathbf{Y}_d$ contains the complex-valued DFT approximating the column m of $\mathbf{X}_d$ in (10), but only over the frequency bins that belong to this octave – outside this octave, $\mathbf{Y}_d$ is zero. The structure of the inverse spectral kernel $\mathbf{V}$ will be described in Sec. 5: we use kernels for which $\mathbf{V} = \mathbf{A}^*$, meaning that the inverse kernel is a conjugate transpose of the forward transform kernel. [3]

Since each column of $\mathbf{Y}_d$ only contains the DFT spectrum for the positive frequencies, each column is augmented with its complex conjugate spectrum to reconstruct the negative frequencies (for real-valued time-domain signals, the DFT spectrum is conjugate-symmetric). The resulting columns are inverse DFT transformed to obtain the time-domain signals within each DFT block, and successive DFT blocks are then overlap-added to construct the entire signal $y_d(n)$ over time.

The signal $y_d(n)$ contains a reconstruction of one octave of the original input signal $x(n)$. This signal is added to a signal that already contains a reconstruction of all the lower octaves $(d+1, d+2, \ldots, D-1)$ in order to obtain a signal $\hat{x}_d(n)$ that approximates the input signal for octaves $d, d+1, \ldots, D-1$. The signal $\hat{x}_d(n)$ is then upsampled by factor two by inserting zeros between the original samples, multiplying the signal by two, and lowpass filtering using zero-phase forward-and-reverse filtering with a sixth-order Butterworth IIR filter having cut-off frequency $f_{\mathrm{s}}/4$ (the same that was used at the analysis stage).

The above process is repeated for each octave at a time, as illustrated in Fig. 5. After reconstructing all the octaves, $d = 0, 1, \ldots, D-1$, the resulting signal $\hat{x}_0(n) \equiv \hat{x}(n)$ is an approximate reconstruction of the input signal $x(n)$.

[3] Note that then $\mathbf{Y}_d = \mathbf{V}^* \mathbf{X}_d^{\mathsf{CQ}} = \mathbf{A}\mathbf{A}^* \mathbf{X}_d$, where multiplication by $\mathbf{A}\mathbf{A}^*$ actually implements a near-perfect one-octave bandpass filter.

The computational complexity of the inverse transform is approximately the same as that of the forward transform: here, instead of FFTs, inverse FFTs are computed, and instead of the spectral kernel, the inverse kernel is applied. Since we use $\mathbf{V} = \mathbf{A}^*$, the inverse kernel is sparse too.

5. KERNEL DESIGN

As already mentioned in Sec. 3.2, we use a transform kernel that contains frequency bins over one octave range, and several time-shifted atoms for each frequency bin k. These time-shifted atoms should cover the input signal over the H_{DFT} samples between the beginning of DFT frame m and the beginning of the next frame, $m + 1$.

5.1 General Considerations

It is natural to expect that all samples of the input signal $x(n)$ have an equal weight in the transform, and therefore to require that successive window functions $w(n)$ for bin k sum up to approximately unity over the entire signal. In the case of an analysis-synthesis system (CQT followed by inverse CQT), the *squares* of successive window functions, $[w(n)]^2$, have to sum to unity. This is because the signal will be windowed twice at the time-frequency location of each atom: once when applying $\mathbf{A}^*$ for the CQT, and second time when applying $\mathbf{V}^* \equiv \mathbf{A}$ for the ICQT. Applying windowing at the synthesis (ICQT) stage is necessary in order to avoid audible artefacts if the signal is manipulated in the CQT transform domain. If no processing takes place in the transform domain, the above requirement (that $[w(n)]^2$ sum to unity) leads to a high-quality reconstruction.

Typically, then, the window function $w(n)$ is defined to be the square root of one of the commonly used window functions (e.g. Hann or Blackman). For analysis-only applications, the square root can be omitted to improve the time-frequency localization properties of the window.

Most window functions (e.g. Hann and Hamming) sum to a constant value only when the distance between successive windows $H_k = \frac{1}{z}N_k$, where $z \geq 2$ is an integer and N_k is the window size for atom k. For an individual frequency bin k, the DFT frame hop H_{DFT} can be chosen to be an integer multiple of $\frac{1}{z}N_k$ so that exactly an integer number of time-shifted atoms would fit between the beginnings of frame m and $m + 1$, and these time-shifted atoms would be stored in the kernel $\mathbf{A}$. However, this requirement for H_{DFT} cannot be simultaneously satisfied for all frequency bins k with different N_k. A reasonable solution is obtained by using a relatively large DFT frame size, in which case H_{DFT} can be made large relative to atom sizes N_k, and thereby H_{DFT} approximately divisible by $\frac{1}{z}N_k$ for all k. This approach leads to a reasonable quality results. However, due to the fact that such a kernel contains a different number of atoms for each bin, accessing and manipulating the CQT coefficients in the transform domain becomes slightly more complicated numerically, and the quality of the reconstruction degrades since neighbouring atoms are not temporally synchronized. Also, sparsity of the spectral kernel $\mathbf{A}$ suffers since DFT frame size is large relative to the atom sizes N_k.

5.2 Proposed Kernel Structure

Due to the above reasons, we propose a kernel, where the atoms *within each octave* are synchronized in the sense that they are centered at the same, successive, points in time. In this case, the temporal ripple can be minimized by using a window function that roughly sums up to unity for a wide range of overlap values. Blackman and Blackman-Harris windows are particularly suitable here: provided that successive windows overlap at least by 66% and 75% for Blackman and B.-Harris windows, respectively, the exact amount of overlap is not important, but successive windows sum up to approximately a constant value which can be normalized to unity. Using such windows it is now possible to use the same number of temporal atoms for each bin by defining a common hop size H_{ATOM} for all atoms in the kernel. The relative hop size H_{ATOM}/N_k will thus vary accross bins k, but only up to the factor two between the smallest and the largest atom in the one-octave kernel.

The parameter H_{ATOM} determines a trade-off between reconstruction quality (SNR of the signal reconstructed by ICQT) and redundancy of the representation. Having small value of H_{ATOM} leads to high quality, but also more redundancy, that is, larger number of CQT coefficients in proportion to the number of samples in the input signal. However, a default value for H_{ATOM} is easy to calculate: we recommend using $H_{\mathsf{ATOM}}/N_K \approx \frac{1}{3}$ or $\frac{1}{4}$, where N_K denotes the lenght of the shortest atom within the one-octave kernel. This leads to redundancy factors around five and quality that is near to the optimal (around 55dB).

It should be noted that although the atoms are centered at same temporal positions within each octave, at the next-lower, octave, the atoms are centered at only every second of these temporal positions, due to the decimation by factor two before processing the next octave.

The number of temporally shifted atoms within the one-octave kernel is the same for all bins k in the above-described approach. In practice, we choose a DFT frame length that is the next power of two of the largest atom within the octave, and then populate the kernel with as many temporally shifted atoms as there fit. The DFT frame hop H_{DFT} is then chosen accordingly. This leads to the most sparse spectral kernel $\mathbf{A}$ and therefore the fastest implementation.

To understand why neighbouring bins sum up to unity over frequency (and not just over time), note that within each octave, neighbouring bins are centered at same point and their lengths N_k are only slightly different, assuming $B > 12$ bins per octave. As a result, the neighbouring bins perform sampling of time-frequency domain that is *locally* very similar to that of the DFT, and similarly to the DFT, sum up to an almost perfectly flat frequency response.

6. REDUNDANCY

The redundancy factor R of the proposed CQT transform is given by

$$R = \frac{2C_{\mathsf{CQT}}}{C_{\mathsf{IN}}} \tag{13}$$

where C_{CQT} and C_{IN} denote the amount of CQT coefficients and the amount of samples in the input signal, re-

spectively. The factor 2 is due to the fact that CQT coefficients are complex-valued.

The amount of CQT coefficients produced by processing the highest octave is $C_{\mathsf{OCT}} = C_{\mathsf{IN}}B/(hN_K)$, where $h = H_{\mathsf{ATOM}}/N_K$ is the atom hop size relative to the length N_K of the shortest atom (hiqhest-frequency bin). Substituting the length N_K from (6), we get

$$C_{\mathsf{OCT}} = \frac{C_{\mathsf{IN}} f_K B(2^{\frac{1}{B}} - 1)}{hqf_{\mathrm{s}}} \approx \frac{0.7 C_{\mathsf{IN}} f_K}{hqf_{\mathrm{s}}} \qquad (14)$$

where the latter approximation is obtained by noting that $B(2^{\frac{1}{B}} - 1) \approx 0.7$ when $B \geq 12$. Here we have assumed that the number of bins per octave $B \geq 12$.

Since the length of the input signal decreases by the factor of two at each decimation, it is easy to see from (14) that the number of CQT coefficients decreases by the same factor for each octave down. Therefore the overall amount of data for a large number of octaves is $C_{\mathsf{OCT}}(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots) \approx 2C_{\mathsf{OCT}}$. Substituting this to (13), the overall redundancy of the CQT transform is

$$R = \frac{2 \times 2 \times C_{\mathsf{OCT}}}{C_{\mathsf{IN}}} = \frac{2.8 f_K}{hqf_{\mathrm{s}}}. \qquad (15)$$

Here we can see that the redundancy is proportional to the highest frequency analyzed, f_K, and inversely proportional to the relative atom hop size h and the Q-value scaling factor q (see 5).

7. INTERFACE TOOLS FOR THE CQT DATA

In the described kernel structure, temporal positions where $X^{\mathsf{CQ}}(k, n)$ is calculated are the same for all bins within one octave (although the actual time resolution of course decreases from the highest to the lowest bin since the atom lengths vary). Moving down to the lower octaves, however, the number of points where $X^{\mathsf{CQ}}(k, n)$ is evaluated decreases by factor two at each octave, and therefore the number of time points where $X^{\mathsf{CQ}}(k, n)$ is evaluated is not the same for all bins from the lowest frequency bin (of the lowest octave) to the highest bin (of the highest octave).

In order to allow the user an easy access to the information without minding the inherent time sampling technique, the reference implementation of the toolbox in Matlab contains interface tools to access the CQT data in a representation that is regularly sampled in time. This "rasterised" CQT data structure is achieved by data interpolation between the time points $X^{\mathsf{CQ}}(k, n)$ that have been computed by the CQT. With the interface tools, the user can obtain the entire CQT matrix representing the input data, or access only extracts of it. It is also possible to access only a certain time slice n of the CQT transform $X^{\mathsf{CQ}}(k, n)$ or all the CQT coefficients of a certain frequency bin over time.

Another important tool is a function for plotting the magnitude of the CQT transform $X^{\mathsf{CQ}}(k, n)$ in a form similar to the DFT spectrogram using the described interpolation technique. Figure 6 shows the CQT transform of a four-second music excerpt containing singing, acoustic guitar, bass, and synthesizer sounds. More examples can be found online at the URL given in Introduction.

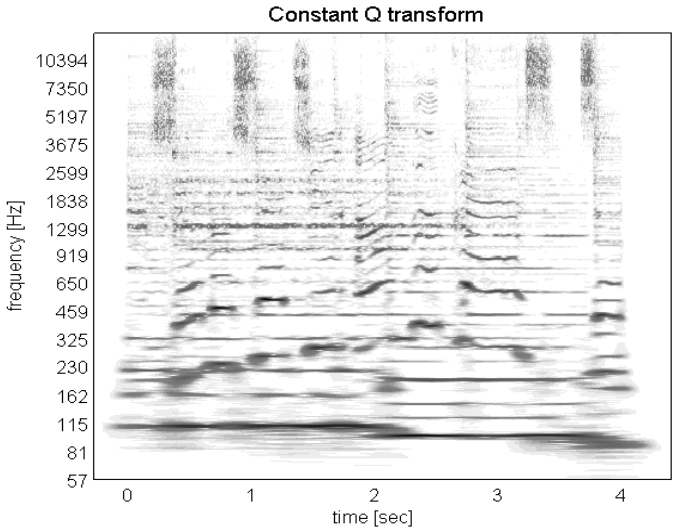


Figure 6. CQT transform of a music excerpt containing singing, acoustic guitar, bass, and synthesizer sounds.

8. RESULTS

Figure 7 shows the quality of the reconstructed time-domain signal $\hat{x}(n)$ as a function of the redundacy R (see (13)) and different window functions $w(n)$. Here the number of bins per octave was $B = 48$. In this plot, the redundancy was increased by decreasing the relative hopsize h of the shortest atom from 0.6 to 0.1. A constant Q scaling factor $q = 1$ has been used, which means that only time-domain redundancy has been added. Using $q = 0.5$ (frequency-domain oversampling) would improve the quality further by $\approx$ 3dB but also increase the redundancy by factor two, therefore results are shown only for $q = 1$.

The input signal was Gaussian random noise, bandpass filtered to contain only frequency components within the range being analysed: we used $f_K = f_{\mathrm{s}}/3 = 14.7$kHz for the highest CQT bin, and analyzed eight octaves down to 57Hz. Random noise represents a "worst case": for music signals, the reconstruction quality is typically a few decibels better. Redundancy factors were calculated by substituting $f_K = 14.7$kHz and $f_{\mathrm{s}} = 2 \times 14.7$kHz into (15), where the latter is the sampling rate required to represent the time domain signal up to 14.7kHz. [4]

Signal-to-noise ratios (SNRs) were calculated by comparing the reconstructed signal $\hat{x}(n)$ after inverse CQT with the input signal $x(n)$:

$$\mathrm{SNR} = 10 \log_{10} \frac{\sum_n [x(n)]^2}{\sum_n [\hat{x}(n) - x(n)]^2} \qquad (16)$$

It can be observed that the choice of the window function has crucial influence on the quality of the reconstruction. For a very low redundancy, corresponding to a large atom hop size, the highest SNR values are achieved using a Hann window. For the redundancy range from 3 to 4.5 the Blackman window performs best, whereas for

[4] Note that if an input signal is to be analyzed up to the Nyquist frequency ($f_K = f_{\mathrm{s}}/2$), the input signal has to be slightly upsampled (say, $f'_{\mathrm{s}} = \frac{4}{3} f_{\mathrm{s}}$) before applying the proposed method, since the lowpass filter $G(f)$ in Fig. 4 is not ideal.

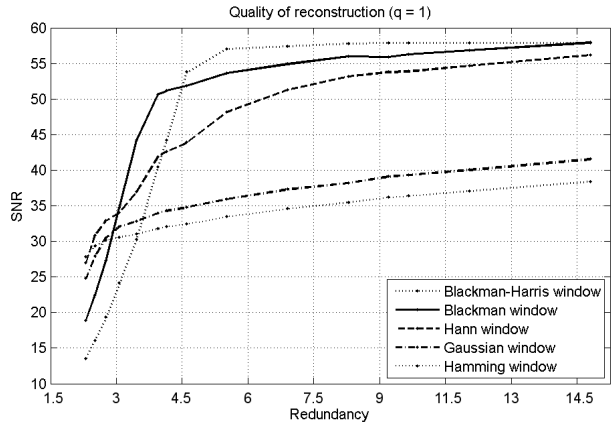


Figure 7. Quality of the reconstructed signal as a function of the window function $w(n)$ and the redundancy R.

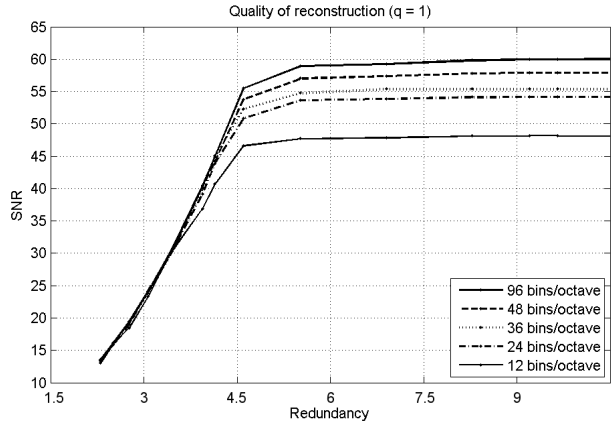


Figure 8. Quality of the reconstructed signal as a function of the number of bins per octave, B, and redundancy R.

$R > 4.5$ the Blackman-Harris window achieves the highest SNR values. This result can be explained by considering the time ripple of the different window functions for varying hop sizes. The Blackman-Harris window shows large time ripple with small overlap values, but for overlap values greater than 75%, consecutive windows sum up to unity almost perfectly. The Blackman window has similar properties but converges slower to a low level of ripple. Figure 7 shows that using Blackman-Harris window, SNR values of about 55dB are achieved with $R \approx 5$.

Fig. 8 shows the quality of the reconstructed time-domain signal $\hat{x}(n)$ as a function of the redundacy R (see (13)) using a Blackman-Harris window and different values for B (bins per octave). It can be observed that the quality of the reconstructed signal improves by increasing the number of bins per octave, achieving up to 60 dB SNR using $B = 96$. The property of the Blackman-Harris window obtaining the highest SNR values already for low redundancy values is independent of the number of bins per octave.

9. ACKNOWLEDGEMENT

Thanks to Wen Xue from Queen Mary University of London for constructive comments on a draft of this paper.

10. CONCLUSIONS

Computationally efficient methods were proposed for computing the CQT and its inverse transform. The proposed techniques lead to a reasonable-quality reconstruction (around 55dB) of an input signal from its CQT coefficients while requiring only moderate redundancy (by factor four or five) in the CQT representation. A reference implementation of the methods is provided as a Matlab toolbox. It is hoped to be useful for several applications, including sound source separation, music signal analysis, and audio effects.

11. REFERENCES

[1] B. C. J. Moore, ed., *Hearing—Handbook of Perception and Cognition*. 2nd ed., 1995.

[2] ISO/IEC, "Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced audio coding (AAC)," Tech. Rep. 13818-7:2006, ISO/IEC, 2006.

[3] S. Mallat, *A wavelet tour of signal processing*. Academic Press, third ed., 2009.

[4] I. Bayram and I. W. Selesnick, "Frequency-domain design of overcomplete rational-dilation wavelet transforms," *IEEE Trans. on Signal Processing*, vol. 57, no. 8, pp. 2957–2972, 2009.

[5] J. Kovacevic and M. Vetterli, "Perfect reconstruction filter banks with rational sampling factors," *IEEE Trans. on Signal Processing*, vol. 41, pp. 2047–2066, 1993.

[6] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U. K. Laine, and J. Huopaniemi, "Frequency-warped signal processing for audio applications," *J. Audio Eng. Soc.*, vol. 48, no. 11, pp. 1011–1031, 2000.

[7] J. J. Burred and T. Sikora, "Comparison of frequency-warped representations for source separation of stereo mixtures," in *121st Audio Engineering Society Convention*, (San Francisco, CA, USA), 2006.

[8] J. C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Am.*, vol. 89, no. 1, pp. 425–434, 1991.

[9] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *J. Acoust. Soc. Am.*, vol. 92, no. 5, pp. 2698–2701, 1992.

[10] D. FitzGerald, M. Cranitch, and M. T. Cychowski, "Towards an inverse constant Q transform," in *120th Audio Engineering Society Convention*, (Paris, France), 2006.

[11] A. Makur and S. K. Mitra, "Warped discrete-Fourier transform: Theory and applications," *IEEE Trans. Circuits and Syst.-I:Fundd. Theor. Appl.*, vol. 48, no. 9, pp. 1089–1093, 2001.

AUTOMATIC MUSIC COMPOSITION BASED ON COUNTERPOINT AND IMITATION USING STOCHASTIC MODELS

Tsubasa Tanaka, Takuya Nishimoto, Nobutaka Ono, Shigeki Sagayama
Graduate School of Information Science and Technology, University of Tokyo
{tanaka,nishi,onono,sagayama}@hil.t.u-tokyo.ac.jp

ABSTRACT

In this paper, we propose a computational method of automatic music composition which generates pieces based on counterpoint and imitation. Counterpoint is a compositional technique to make several independent melodies which sound harmonious when they are played simultaneously. Imitation is another compositional technique which repeats a theme in each voice and associate the voices. Our computational method consists of the stochastic model of counterpoint and that of imitation. Both stochastic models are simple Markov models whose unit of state is a beat. We formulate the problem as the problem to find the piece which maximize the product of probabilities that correspond to both stochastic models. Dynamic programming can be used to find the solution because the models are simple Markov models. Experimental results show that our method can generate pieces which satisfy the requirements of counterpoint within two successive beats, and can realize imitations of the theme with flexible transformations.

1. INTRODUCTION

Counterpoint is one of the most basic principles for composition and arrangement. It is a composition technique for tuning voices, which are independent in contour and rhythm. Counterpoint consists of many prohibitions and recommendations such as ones which regulate the progression of a melody or the progression of interval between voices as shown in figure 1. It takes long time to master counterpoint, because finding melodies which satisfy many conditions is difficult. Therefore, realization of automatic contrapuntal composition will be valuable.

Imitation is also a very important technique for contrapuntal music. In a piece based on imitation, a theme melody is exposed at the start of the piece and melodies which imitate the theme repeatedly occur in all voices. As a result, imitation gives sense of unity to the piece. Imitation is an indispensable basis for some musical forms such as canon or fugue. Automatic composition systems based on counterpoint and imitation are useful because whole musical pieces can be obtained by simply providing a short theme and setting the structure of the piece.

Figure 1. Examples of requirements of counterpoint. The left figure shows a requirement for melodies. Conjunct motions (bar 1) should be used frequently. Skips (bar 2) should be used occasionally. The right figure shows a requirement for intervals between voices. Parallel fifths is prohibited [1, 2].

There are some previous studies about automatic counterpoint, such as rule-based approaches [3, 4] and methods based on stochastic models [5, 6]. A method based on hidden Markov models is also proposed in [7]. These studies are dealing with a kind of arrangement which is a composition of counter-melodies played simultaneously with a main melody which is given (cantus firmus).

In this paper, we aim at contrapuntal composition based on imitation of a given theme. We do not aim at generating pieces which are better than human compositions, but generating pieces which are acceptable from the standpoint of counterpoint and imitation.

Cope also proposes an contrapuntal composition system , which is based on the method of re-combination using the fragments of existing music as the components of generated pieces [8]. The limitation of his approach is that results are likely to be similar to particular pieces he uses, and that is his intention. The difference between our purpose and Cope's purpose is on this point. We intend to formulate general model without such limitation of the method of re-combination and generate new pieces.

We focus on two-voice free counterpoint, while there are various types in rhythms and the number of voices in counterpoint. Although the number of voices varies, the most important things are the relations in each pair of voices. Therefore, we focus on two-voice counterpoint, which is the most basic type of counterpoint about the number of voices. Concerning the rhythm, there are strict counterpoint and free counterpoint. The former is developed for pedagogy and has several kinds of fixed rhythms. The latter has variable rhythms and is more practical. We focus on practical composition and therefore deal with free counterpoint. In the following sections, counterpoint means free counterpoint.

2. STOCHASTIC APPROACH TO COUNTERPOINT AND IMITATION

In this section, we discuss a stochastic approach to the process of composition and introduce three assumptions. The

formulation for the automatic composition based on counterpoint and imitation is derived from these assumptions.

2.1 Asummption about the Contrapuntal Composition

To generate proper contrapuntal pieces, we should define what is proper and what is not proper for the music we try to generate. For this purpose, it is considered to be effective to adopt a stochastic approach. Although we can use the knowledge of counterpoint represented by explicit rules, all the qualities of music can not be represented only by rules of counterpoint. In actual composition, a substantial part of selecting notes are handled by the composer's intuition. Therefore, it is better to model not only the knowledge of counterpoint but also the tendencies of composers. To model composer's tendencies of composing contrapuntal pieces, we propose a stochastic approach based on the following assumption;

Assumption 1 When an experienced composer write a contrapuntal piece, he or she is subjected to a probability distribution $\Pr(X)$ which is formed by knowledge and training of counterpoint, and realizes composition by a trial from the probability distribution (the variable X corresponds to a piece of music).

In this point of view, existing contrapuntal pieces can be considered as the outputs from the probability distribution $\Pr(X)$ and have high probabilities. In reverse, pieces which have high probabilities are considered to satisfy the requirements of counterpoint and to be acceptable as pieces of music. This probability distribution is considered to be the model of the contrapuntal composer.

2.2 Assumption about Three Steps of Composition based on Imitation

The process of composition based on imitation can be roughly divided into three steps:

1. The first step is to obtain a theme T. It can be composed by composers or taken from existing melodies.
2. The second step is to plan S, the structure of imitations and cadences. The structure of imitations indicates where and from what pitch imitations begin.
3. The third step is to select concrete notes and compose the actual piece X.

In practice, these three steps are not necessarily separated. However, these steps are considered to be carried out step by step from an idealistic viewpoint. S varies with the length and proportion of the piece according to time and circumstances. Therefore, S is better to be planned or selected by the user of the system before the automatic composition is started. For this reason, S should be also given as well as the theme T. These ideas are summed up as the following assumption;

Assumption 2 When a composer makes a piece based on imitation, he or she firstly obtain T, a theme to be imitated. Next, S, the structure of imitations and cadences is decided. Then, composition using the theme T and structure S is started.

If we deal with the style of tonal counterpoint, the structure S should include the plan of code progressions and key modulations. In this paper, however, S do not includes them. To make the problem simple, we deal with modal counterpoint, which is mainly the style of the Renaissance period and do not have code progressions.

2.3 Asummption about the Composition based on Imitation

The process of composition based on imitation of a theme can be viewed from the standpoint of probability in a similar way to assumption 1. Although imitation tends to be similar to the theme, imitation is not necessarily identical to it. For example, pitch shifts on the scale, intention to avoid unnatural harmonies, and tonality often affect imitations and transform them from the original theme. Such flexibility is necessary where strict imitations cause unfavorable results. A probability distribution in the next assumption is useful to realize such flexibility;

Assumption 3 When an experienced composer composes a piece based on imitation of a theme, he or she obtains the theme T and plans the structure S. After that, the composer composes the piece X. At this time, the composer is subjected to a conditional probability $\mathrm{Im}(X|T,S)$, which is formed by experiences of composition.

This probability $\mathrm{Im}(X|T,S)$ is considered as the model of imitation-based composition.

2.4 Formulation Based on the Three Assumptions

From these three assumptions, the problem of the automatic contrapuntal composition based on imitation of a theme is considered as a problem to obtain the piece X which give high probabilities to both the probability of the stochastic model of counterpoint and that of imitation. We show an example of the piece we intend to generate in figure 2. When T and S are given, the best piece $\tilde{X}$ can be formulated as the piece X which maximizes the product of $\Pr(X)$ and $\mathrm{Im}(X|T,S)$:

$$\tilde{X} = \operatorname*{argmax}_X \Pr(X)\mathrm{Im}(X|T,S). \quad (1)$$

By this formulation, pieces which realize flexible imitation and satisfy counterpoint are expected to be generated. However, it is difficult to obtain the values of $\Pr(X)$ and $\mathrm{Im}(X|T,S)$ directly from statistics on exsisting pieces of music. The reason is that the number of possible pieces is much larger than that of existing pieces. Therefore, it is necessary to extract the essential information that determine $\Pr(X)$ and $\mathrm{Im}(X|T,S)$, and to approximate these probabilities.

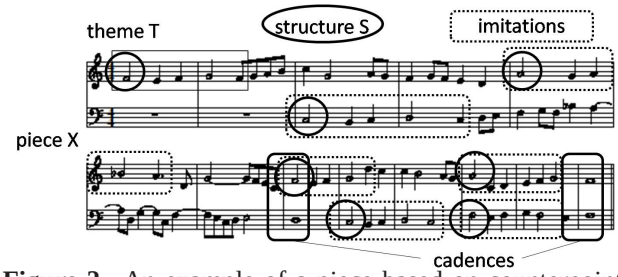


Figure 2. An example of a piece based on counterpoint and imitation [2]. Black circles indicate the starting points and the starting pitches of imitations. Domains bounded by dashed lines indicate imitations.

3. FORMALIZATION OF STOCHASTIC MODELS FOR COUNTERPOINT AND IMITATION

In this section, we propose stochastic models of counterpoint and imitation which approximate equation 1 in the case of two voices.

3.1 Formalization of the Stochastic Model of Counterpoint

To calculate $\Pr(X)$ and find $\tilde{X}$, data sparseness problem and exponential expansion of computational cost must be dealt with. A realistic solution is to approximate the probability by using existing probabilities and transition probabilities of short unit. Meanwhile, in composition and listening music, there are beat transitions behind concrete notes and they are written or listened upon beat transitions. Therefore, it is natural to adopt the length of a beat as the unit length of the states of probabilities. Defining $x_1, x_2, \quad , x_N$ as information of each beat of the piece X, $\Pr(X)$ is transformed as:

$$\Pr(X) = \prod_{i=2}^{N} \Pr(x_i | x_{i\ \ 1}, x_{i\ \ 2}, \quad , x_1) \Pr(x_1). \quad (2)$$

Equation (2) can be approximated using a simple Markov model, which is a favorable model in perspective of the computational cost. The ideas behind the approximation are as follows. Most requirements of the counterpoint are concerned with regulating the transition of pitch and the interval between voices. These requirements tend to be included within two successive beats. Therefore, assuming that long-raged dependences are ignorable, $\Pr(X)$ can be approximated by a simple Markov model whose unit of the state is a beat as:

$$\Pr(X) \simeq \prod_{i=2}^{N} \Pr(x_i | x_{i\ \ 1}) \Pr(x_1). \quad (3)$$

Dynamic programming, by which the solution of probability maximization can be efficiently searched, can be applied to the simple Markov model. That is why the simple Markov model is advantageous. The details of dynamic programming are explained in section 4.

It is difficult to obtain $\Pr(x_i | x_{i\ \ 1})$ and $\Pr(x_1)$ statistically, because x_i contains information of multiple voices and have many possible variations of states, and therefore further approximation is necessary. Referring to the requirements of counterpoint, it is considered that $\Pr(x_i | x_{i\ \ 1})$ and $\Pr(x_1)$ are correlated with some elements which consist of them. As such elements, there are;

Transition of pitch

Transition of rhythm in each voice

Transition of interval between voices

Co-occurrence of rhythms of both voices

It is considered that there is a tendency that the higher probabilities of these elements are, the higher the probability $\Pr(x_i | x_{i\ \ 1})$ and $\Pr(x_1)$ are. The transition of pitch in a voice is related to the composition of the melodic progression, the transition of rhythm in a voice is related to the sense of rhythm and the note lengths, and the transition of interval between voices are related to the composition of harmony. Co-occurrence of rhythms of both voices are related to independence of voices and balance between voices.

Therefore, in equation (3), it is appropriate that $\Pr(x_i | x_{i\ \ 1})$ and $\Pr(x_1)$ are approximated by products of probabilities related to such elements;

$$\begin{aligned} \Pr(X) \simeq \prod_{i=2}^{N} \{ & \Pr(p_i^1 | p_{i\ \ 1}^1) \Pr(r_i^1 | r_{i\ \ 1}^1) \\ & \Pr(p_i^2 | p_{i\ \ 1}^2) \Pr(r_i^2 | r_{i\ \ 1}^2) \\ & \Pr(a_{i,i\ \ 1}) \Pr(r_i^1, r_i^2) \ \} \\ & \Pr(p_1^1) \Pr(p_1^2) \Pr(r_1^1) \Pr(r_1^2) \Pr(a_1) \Pr(r_1^1, r_1^2). \end{aligned} \quad (4)$$

r_i^j and p_i^j are the information of the rhythm pattern and the series of pitch of each voice in the beat i. j means the index of the part. $j = 1$ corresponds to the upper voice and $j = 2$ corresponds to the lower voice. $a_{i,i\ \ 1}$ is the series of transition of the interval between the voices from the last interval of $x_{i\ \ 1}$ to the last interval of x_i (the information of motions are also included. An oblique motion from major third to perfect fourth is an example of a component of the series). a_1 is the series of transition of interval in x_1.

3.2 Formalization of the Stochastic Model of Imitation

When we try to obtain the value of $\mathrm{Im}(X|T, S)$, data sparseness problem and the problem of computational cost occur as well as when we obtain the value of $\mathrm{P}(X)$. To deal with these problem, it is necessary to approximate $\mathrm{Im}(X|T, S)$. Before formalizing the approximated stochastic model of imitation, we define some variables. Q is defined as the number of imitations. T_n is defined as what T is shifted in pitch to start with the first pitch of the nth imitation. The unit of pitch shift is a semitone. The first pitch of the nth imitation is determined by S. M_n is the part of X which corresponds to the nth imitation. L is the number of beats within T (which is equal to the number of beats within T_n or M_n). t_l^n is information of the lth $(1 \leq l \leq L)$ beat of T_n. m_l^n is information of the lth beat of M_n.

In view of following ideas, we approximate $\mathrm{Im}(X|T, S)$. We can consider that there is a tendency that the more

M_n, which is a part of X, is similar to T_n, the higher the probability $\mathrm{Im}(X|T,S)$ is. Also, it is natural to assume that defferent imitations are independent. On the other hand, it is considered that whatever the notes in the region which is not the part of imitations or cadences are, $\mathrm{Im}(X|T,S)$ do not vary so much and can be regarded as constant. To make the problem simple, we always treat X as a piece which have the cadences determined by S (otherwise, $\mathrm{Im}(X|T,S)$ is regarded as zero). Therefore, $\mathrm{Im}(X|T,S)$ can be approximated as the product of the probabilities of each imitation as;

$$\mathrm{Im}(X|T,S) \simeq \prod_{n=1}^{Q} \mathrm{Im}(M_n|T_n). \tag{5}$$

To maximize both probabilities of the stochastic models at once, it is advantageous if $\mathrm{Im}(M_n|T_n)$, the probability of imitation of each time in equation (5), can be represented by a simple Markov model which have the unit of a beat, similar to the contrapuntal stochastic model. If $\mathrm{Im}(M_n|T_n)$ is represented by a simple Markov model, dynamic programming can be used.

$\mathrm{Im}(M_n|T_n)$ is considered to depend on the similarity of T_n and M_n. This similarity is considered to be determined mainly by four elements discussed later in this subsection. These four elements are reflected in $\mathrm{Im}(m_l^n|m_{l\ 1}^n, t_l^n, t_{l\ 1}^n)$, which is a part of $\mathrm{Im}(M_n|T_n)$. $\mathrm{Im}(M_n|T_n)$ can be transformed as equation (6). The relations other than that of the two successive beats in each probability can be ignored as we mention later in this subsection. Therefore equation (7) can be derived from equation (6).

$$\begin{aligned}
&\mathrm{Im}(M_n|T_n) \\
&= \prod_{l=2}^{L} \mathrm{Im}(m_l^n|m_{l\ 1}^n \quad m_1^n, T_n)\mathrm{Im}(m_1^n|T_n) \quad (6)\\
&\simeq \prod_{l=2}^{L} \mathrm{Im}(m_l^n|m_{l\ 1}^n, t_l^n, t_{l\ 1}^n)\mathrm{Im}(m_1^n|t_1^n) \quad (7)
\end{aligned}$$

Equation (7) indicates that dynamic programming can be applied, because the stochastic model of imitation is equivalent to the simple Markov model in this equation.

As previously mentioned, the elements which are important in order to measure the similality of T_n and M_n are:

1. similarity of direction of pitch transition (upward and downward skip, upward and downward conjunct motion, and stay in the same pitch)
2. similarity of melodic interval
3. similarity of pitch
4. similarity of rhythm.

Among the four elements, the first and the forth are considered to be most important because the rough character of the melody is determined by these. The second and the third elements are considered to have a secondary role. If these four similarities between T_n and M_n are high enough, the imitation is expected to be successful.

Similarity of direction and melodic interval can be judg ed by comparison of the transitions from the previous note to the present note. Similarity of pitch can be judged by comparison of only the present note. Similarity of rhythm can be judged by comparison of the onset in each time. Therefore, these four elements are considered to be reflected in $\mathrm{Im}(m_l^n|m_{l\ 1}^n, t_l^n, t_{l\ 1}^n)$.

4. DYNAMIC PROGRAMMING

In this section we explain dynamic programming, which is a very efficient algorithm and can be used to find the X that maximizes the probability. Generally speaking, it takes $\mathrm{O}(c^N)$ computing time to search the best answer for series of length N. However, dynamic programming can reduce the computing time to $\mathrm{O}(N)$ by using the locality of a Markov model.

In this paper, the application of dynamic programming is as follows. We define $\mathrm{p}(x_i, x_{i\ 1})$ as the product of the probabilities across the beat $i-1$ and the beat i relating to both stochastic models of counterpoint and imitation (eqation (4) and (7)). x_i is the information of the beat i in X. $P_{\max}$ is the maximum of the cumulative probability of all $\mathrm{p}(x_i, x_{i\ 1})$ for each i. $P(x_i)$ is the maximum cumulative probability of $\mathrm{p}(x_i, x_{i\ 1})$ until x_i appears at the beat i. The Markov property enable us to represent $P(x_i)$ recursively as;

$$P(x_i) \quad = \quad \max_{x_{i-1}} \{p(x_i, x_{i\ 1})P(x_{i\ 1})\}. \tag{8}$$

By preserving $P(x_{i\ 1})$ for every $x_{i\ 1}$, we can obtain $P(x_i)$ sequentially. Finally, we can obtain $P_{\max}$ as the maximum of the $P(x_N)$ for every $P(x_N)$. Furthermore, preparing the backward pointer $b(x_i)$ which indicates the optimal path for x_i from beat $i-1$, we can obtain the optimal path which realizes $P_{\max}$. The optimal path can be obtained by going back from the x_N which realize $P_{\max}$ at the final beat to the first beat. $b(x_i)$ is represented as;

$$b(x_i) \quad = \quad \operatorname*{argmax}_{x_{i-1}} \{p(x_i, x_{i\ 1})P(x_{i\ 1})\}. \tag{9}$$

5. EXPERIMENT

5.1 Conditions of Experiment

In this section, we report the experiment of generating musical pieces by the proposed method.

Before computing with the algorithm, we manually produced the theme T which has length of 2 bars at the start of the lower voice. We also produced the structure S which start imitation at the 2nd , 8th, and 13th bars with the note B in the upper voice, and the 7th and 12th bars with the note E in the lower voice. Cadences, which are included in S, were also given manually.

In relation to the configuration of the probabilities, further approximations were done as described in the following subsections. As the number of notes in a beat increases,

the number of states also increases. When the number of states is large, necessary amount of statistical data is also large. In general, further approximation of equation (4) and (7) which make it easy to set the value statistically is necessary. To determine the values of the probabilities, musical knowledge was also used. Statistical features were taken from "*Invention*" by J.J.Fux, which is shown in the textbook of counterpoint [2] as one of the example of a piece based on counterpoint and imitation.

We adopted eighth note as the minimum unit of time value of a note, and quarter rest as the minimum unit of a rest. Rests were not used except in the parts of exposition of the theme. Dynamic programming was used to find the results.

5.2 Transition Probability of Pitch

Considering that local properties are the most important in counterpoint, it is reasonable to approximate the transition probabilities of pitch $\Pr(p_i^j|p_{i\ \ 1}^{j})$ by the product of local probabilities. We approximated $\Pr(p_i^j|p_{i\ \ 1}^{j})$ by the product of unigram probabilities of each transition of pitch from the last pitch of $p_{i\ \ 1}^{j}$ to the last pitch of p_i^j. Unigram probabilities are represented by relative intervals of transition of pitch (such as major third) to deal with data sparseness problem. Correction by geometric mean of unigram probabilities is adopted as equation (10), not to be effected by the number of notes:

$$\Pr(p_i^j|p_{i\ \ 1}^{j}) \simeq \begin{cases} \prod_{k=1}^{N(P_{i,i-1}^{j})} \Pr(p_{i,i\ \ 1,k}^{j})^{\frac{1}{N(P_{i,i-1}^{j})}} \\ \quad \text{if } N(p_{i,i\ \ 1}^{j}) \neq 0 \\ P_0 \quad \text{if } N(p_{i,i\ \ 1}^{j}) = 0 \end{cases} \tag{10}$$

$p_{i,i\ \ 1}^{j}$ is the series of transitions of pitch. $p_{i,i\ \ 1,k}^{j}$ is the kth transition of pitch in $p_{i,i\ \ 1}^{j}$. $N(p_{i,i\ \ 1}^{j})$ is the number of transitions of pitch. Where the number of times of pitch transition is zero (long notes such as half note, ties, etc.), $\Pr(p_i^j|p_{i\ \ 1}^{j})$ is given by P_0. P_0 is obtained by statistics as the number of times there is no change of pitch within two successive beats, divided by the number of times there are changes of pitch within two successive beats. This division is done to balance lower case with the upper case in equation (10). Normalization is not done, because it does not affect the result. $\Pr(p_1^j)$, which corresponds to the first beat, was approximated within a beat in a similar way.

When the statistics of the unigram probabilities of pitch transition are taken, the probabilities are regarded as the same value whether the transitions are upward or downward and whether the intervals are major or minor. Where the interval is compound interval over augmented eighth, it is regarded as the simple interval (for example, minor tenth is regarded as minor third). If violations of counterpoint (such as succession of skips in the same direction) occur in two successive beats, the corresponding probabilities are lowered in equation (10). The unigram probabilities of pitch transition are shown in table 1.

In the piece from which the statistics are taken, no interval of sixth is included by chance. To correct the value of probability of sixth interval, we regarded the value of probability of sixth interval as the same value as that of fifth interval. It is reasonable that the probabilities of prime, seventh and octave intervals are zero in table 1. That is because prime interval is not favorable and larger skips than seventh interval are prohibited in counterpoint. The transitions of pitch in the regions of imitation are counted only once, because the same patterns which are unique to the theme appear many times.

Table 1. Unigram probabilities of pitch transition.

interval of pitch transition	$\Pr(p_{i,i\ \ 1,k}^{j})$
prime	0.000
second	0.824
third	0.095
fourth	0.054
fifth	0.014
sixth	0.014
seventh	0.000
octave	0.000

5.3 Transition Probability of Interval

We approximate transition probabilities of interval in two steps, because there are many combinations in transitions of interval.

In the first step, we approximate $\Pr(a_{i,i\ \ 1})$ by the product of the transition probability of interval $\Pr(b_{i,i\ \ 1})$ and the probability of motion $\Pr(c_{i,i\ \ 1})$ on the assumption that both are independent:

$$\Pr(a_{i,i\ \ 1}) \simeq \Pr(b_{i,i\ \ 1})\Pr(c_{i,i\ \ 1}). \tag{11}$$

$b_{i,i\ \ 1}$ is the series of intervals included in $a_{i,i\ \ 1}$ and $c_{i,i\ \ 1}$ is the series of motions included in $a_{i,i\ \ 1}$.

In the second step, $b_{i,i\ \ 1}$ and $c_{i,i\ \ 1}$ are approximated in a similar way to the case of the transition probability of pitch, and the correction by geometric mean were done as equation (12) and (13):

$$\Pr(b_{i,i\ \ 1}) \simeq \begin{cases} \prod_{k=2}^{N(b_{i,i-1})} \Pr(b_{i,i\ \ 1,k}|b_{i,i\ \ 1,k\ \ 1})^{\frac{1}{N(b_{i,i-1})}} \\ \quad \text{if } N(b_{i,i\ \ 1}) \neq 0 \\ B_0 \quad \text{if } N(b_{i,i\ \ 1}) = 0 \end{cases} \tag{12}$$

$$\Pr(c_{i,i\ \ 1}) \simeq \begin{cases} \prod_{k=1}^{N(c_{i,i-1})} \Pr(c_{i,i\ \ 1,k})^{\frac{1}{N(c_{i,i-1})}} \\ \quad \text{if } N(c_{i,i\ \ 1}) \neq 0 \\ C_0 \quad \text{if } N(c_{i,i\ \ 1}) = 0 \end{cases} \tag{13}$$

$b_{i,i\ \ 1,k}$ is the kth interval of the series $b_{i,i\ \ 1}$. $N(b_{i,i\ \ 1})$ is the number of the transitions of interval in the series $b_{i,i\ \ 1}$. $c_{i,i\ \ 1,k}$ is the kth motion of the series $c_{i,i\ \ 1}$. $N(c_{i,i\ \ 1})$ is the number of motions in the series $c_{i,i\ \ 1}$. Where the number of times of transition of interval is zero, the probability is given by B_0. B_0 is obtained by statistics as the number of times there is no change in interval within two successive beats, divided by the number of times where there are changes of interval within two successive beats. Where the

Table 2. Bigram transition probabilities of interval.

before \ after	perfect	imperfect consonant	dissonant
perfect	0.10	0.40	0.50
imperfect consonant	0.36	0.36	0.27
dissonant	0.09	0.91	0.00

Table 3. Unigram probabilities of motion.

motion	$\Pr(c_{i,i\ \ 1,k})$
parallel	0.121
contrary	0.212
oblique	0.667

Table 4. The values of P_0,B_0,C_0.

P_0	0.342
B_0	0.050
C_0	0.050

number of times of motion is zero, the probability is given by C_0. C_0 is obtained by statistics as the number of times there is no motion within two successive beats, divided the number of times there are motions within two successive beats. $\Pr(a_1)$, which corresponds to the first beat, is approximated within a beat in a similar way.

Intervals of the same category (perfect, imperfect consonant, and dissonant) are identified in statistics to reduce the varieties of transition of interval. If violations of counterpoint occur in two successive beats, the corresponding probabilities are lowered in equation (11). The bigram transition probabilities of interval are shown in table 2 and the unigram probabilities of motion are shown in table 3. The values of P_0, B_0, and C_0 are shown in table 3.

5.4 Co-occurrence Probability of Rhythms

Co-occurrence probabilities of rhythms can be determined by inference based on musical knowledge. In counterpoint, there is a tendency that rhythms continue regularly by placing onset in either voice. Where there is a onset in the head of r_i^1 or r_i^2, $\Pr(r_i^1, r_i^2)$ is considered to be high. By such inference, $\Pr(r_i^1, r_i^2)$ is determined as:

$$\Pr(r_i^1, r_i^2) \simeq \begin{cases} 0.9 & \text{if there is the onset on the head of } r_i^1 \text{ or } r_i^2 \\ 0.1 & \text{if there is not the onset on the head of } r_i^1 \text{ and } r_i^2 \end{cases} \quad (14)$$

Possible rhythm patterns for a beat within the condition of the experiment are shown in figure 3. The ties in figure 3 are the extension of duration to the next beat or from the previous beat.

5.5 Transition Probabilities of Rhythm

Transition probabilities of rhythm $\Pr(r_i^j | r_{i\ \ 1}^j)$ are determined by statistics from the piece mentioned above, because the transition probabilities of rhythm can not be determined by a simple rule.

There are nine possible rhythms in the condition of this experiment as shown in figure 3. The transition probabilities among them are shown in table 5. Each possible $\Pr(r_1^j)$, which corresponds to the first beat, is assigned the same value.

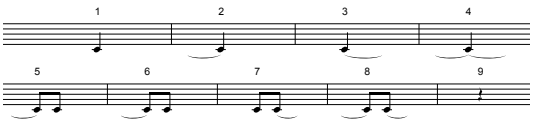


Figure 3. Possible rhythm patterns in the condition of the experiment.

Table 5. Transition probabilities of rhythm.

rhythm pattern befeore \ after	1	2	3	4	5	6	7	8	9
1	0.441	—	0.382	—	0.176	—	—	—	—
2	0.591	—	0.182	—	0.091	—	—	—	0.136
3	—	0.767	—	0.033	—	0.200	—	—	—
4	—	0.500	—	0.500	—	—	—	—	—
5	0.267	—	0.267	—	0.467	—	—	—	—
6	—	—	1.000	—	—	—	—	—	—
7	—	—	—	—	—	—	—	—	—
8	—	—	—	—	—	—	—	—	—
9	0.500	—	0.500	—	—	—	—	—	—

5.6 Probability of Imitation

It is difficult to take statistics of $\mathrm{Im}(m_l^n |\, m_{l\ \ 1}^n, t_l^n, t_{l\ \ 1}^n)$ in equation (7) because the probability depends on the theme and it has many parameters. On the other hand, $\mathrm{Im}(m_l^n |\, m_{l\ \ 1}^n, t_l^n, t_{l\ \ 1}^n)$ can be determined by inference based on musical knowledge. Considering that an imitation tend to be similar to the theme, there should be a tendency that where the similarity between $(m_l^n, m_{l\ \ 1}^n)$ and $(t_l^n, t_{l\ \ 1}^n)$ is high, $\mathrm{Im}(m_l^n | m_{l\ \ 1}^n, t_l^n, t_{l\ \ 1}^n)$ is also high.

To measure the similarity between the theme and the imitation, the four elements mentioned in section 3 (direction, melodic interval, pitch, and rhythm) can be used. Using the four elements, the similarities $-\Delta_{l,n}^k (k = 1, 2, 3, 4)$ are defined as follows. To compare $(m_l^n, m_{l\ \ 1}^n)$ and $(t_l^n, t_{l\ \ 1}^n)$, these are sliced into the length of eigths note, which is the minimum unit in this experiment.

We define similarity of direction as $-\Delta_{l,n}^1$. $\Delta_{l,n}^1$ is the sum of the absolute values of the differences of direction between each slice of $(m_l^n, m_{l\ \ 1}^n)$ and $(t_l^n, t_{l\ \ 1}^n)$. The sum is taken from the last slices of $m_{l\ \ 1}^n$ and $t_{l\ \ 1}^n$ to the last slices of m_l^n and t_l^n. Directions are represented as follows. the upward and downward skips are represented as ± 3 . Upward and downward conjunct motions are represented as ± 2. Continuation of the same pitch is represented as 0.

Similarity of melodic interval is defined as $-\Delta_{l,n}^2$. $\Delta_{l,n}^2$ is defined as the sum of the absolute values of the differences of melodic intervals (the unit is semitone). The sum is taken from the last slices of $m_{l\ \ 1}^n$ and $t_{l\ \ 1}^n$ to the last slices of m_l^n and t_l^n.

Similarity of pitch is defined as $-\Delta_{l,n}^3$. $\Delta_{l,n}^3$ is the sum of the absolute values of the differences of pitch (the unit is a semitone). The sum is taken from the first slices to the last slices of m_l^n and t_l^n.

Similarity of rhythm is defined as $-\Delta_{l,n}^4$. $\Delta_{l,n}^4$ is the sum of the absolute values of the differences of onset values. Onset value is defined as 1 for the position where the onset exist, and 0 for the position where the onset do not exist. The sum is taken from the first slices of m_l^n and t_l^n to the last slices of m_l^n and t_l^n.

Concerning $\Delta_{l,n}^1$ and $\Delta_{l,n}^2$, the absolute values of the differences is not included in the sum where the transition

of pitch or interval is interrupted by a rest. Concerning $\Delta^3_{l,n}$, the absolute values of the differences is not included in the sum where rests appear.

Using these four similarities, $\mathrm{Im}(m_l^n|m_{l\ 1}^{n}, t_l^n, t_{l\ 1}^{n})$ are represented as:

$$\mathrm{Im}(m_l^n|m_{l\ 1}^{n}, t_l^n, t_{l\ 1}^{n}) \simeq \exp(-\prod_{k=1}^{4} \lambda_k \Delta^k_{l,n}). \quad (15)$$

$\lambda_k (k = 1, 2, 3, 4)$ in this equation are the weights of each λ_k, which are tuned manually in preliminary trial runs. Values from λ_1 to λ_4 are set as $12.0, 5.5, 0.5, 40.0$ respectively. $\mathrm{Im}(m_1^n|t_1^n)$, which corresponds to the first bar, is similarly approximated within a beat.

5.7 Result and Discussion of Experiment

From the observation of the results, it is confirmed that the proposed method can generate pieces which largely satisfy the requirements of counterpoint and imitation. As an example, one of the generated pieces is shown in figure 4.

Basic prohibitions of counterpoint such as parallel fifth or hidden fifth, which are included within succsessive two beats are not occurring in the piece shown in Figure 4. In the bar 5, 6, and 15, suspensions, which is important for counterpoint are occurring and dissonances are resolved appropriately. Both melodies are independent in the standpoint of rhythms in both voices at a time and the motions between both voices. However, there are some violations of counterpoint. In the bar 3, octave intervals appear both in the beat 2 and the beat 4. This is a prohibition of counterpoint which is called "successive accented perfect fifths or octaves" [2]. Such violations are sometimes observed. The reason is considered to be that stochastic model of counterpoint is approximated by simple Markov model whose states have the unit of beat and this model can not deal with the requirements of counterpoint which have the length over more than two beats. To cope with this problem, re-evaluation of the N-best solutions might be effective. From the N-best solutions, one which completely satisfy the requirements of counterpoint might be found.

Concerning imitation, adequate treatment is realized by the effect of stochastic model of imitation. In the positions where a imitation is done only in one voice (such as the imitation from the bar 3) the theme is not be transformed. On the other hand, in the positions where imitations in both voices are overlapping (such as imitation from the bar 13.) the theme is slightly transformed. If strict imitation is done there, both imitations will interfere mutually and cause violations of counterpoint. From such observation, we can say that the stochastic model of imitation realizes flexible imitation where transformations are necessary.

5.8 Generation with Given Themes

In the previous subsection, the theme is given by the authors. In this subsection, we give examples of the results in which we use well-known melodies for the theme to demonstrate that the proposing method can generate pieces without user's modification to the theme which may give advantage unfairly to the proposed method.

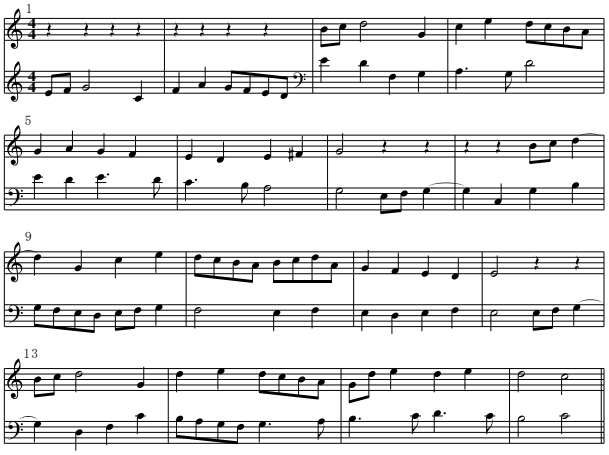


Figure 4. An example of the result. The lower voice of the first two bars is the theme.

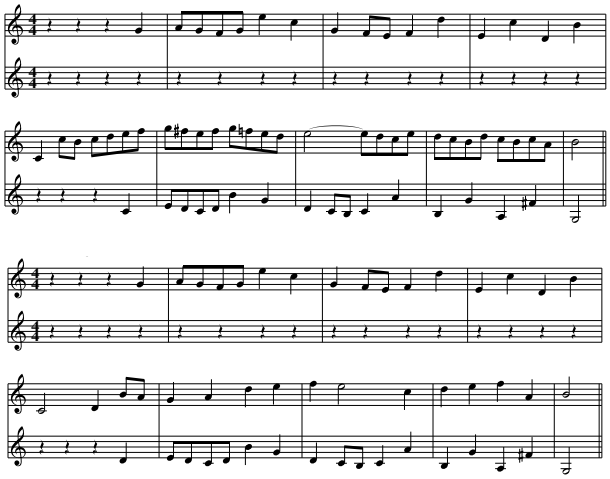


Figure 5. The beginning of the fugue of J.S.Bach (upper) and the result generated with the same theme (lower).

Figure 5 shows the result using the same theme as the fugue of "The Well-Tempered Clavier Vol.1 No.3" by J.S. Bach. In this result, counterpoint is well satisfied and the imitation is also done well. However, in this theme there are skips with seventh interval, which is a prohibition of counterpoint. In the actual pieces like this, violations are sometimes done deliberately. To deal with such cases, we did a smoothing by setting the probability of skip with seventh interval as 0.001, not 0.

Figure 6 shows the result using the melody of "Little Hans" for the theme. In this result, successive accented perfect fifths or octaves are occurring across three successive beats in the bar 9, 17, and 26. In addition, repetitions of a phrase, which are unfavorable, are occurring from the bar 26 to 28. This is also caused by the approximation by a simple Markov model. If we can deal with this weak point by a method such as re-evaluation of the N-best solutions, acceptable results might be generated.

6. CONCLUSION

We proposed stochastic models for an automatic composition based on counterpoint and imitation. The solution is obtained as a musical piece which maximizes the product

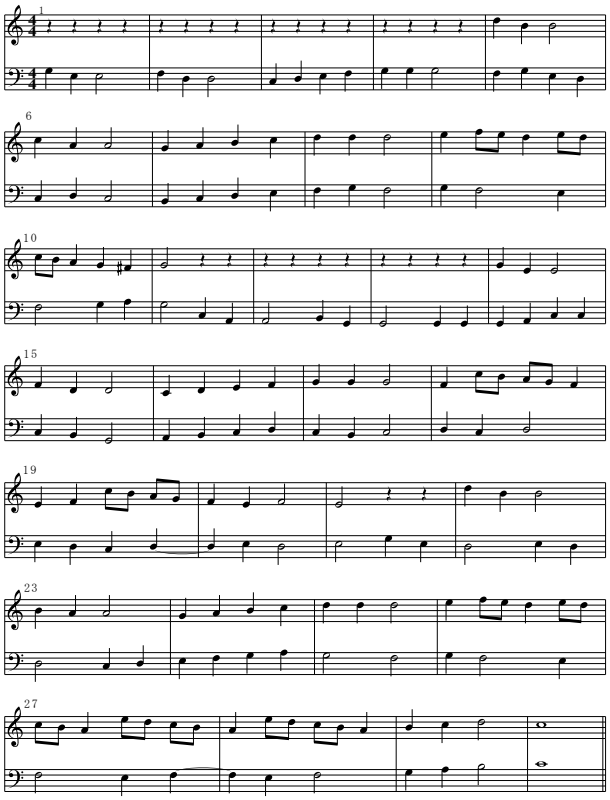


Figure 6. A result generated with the theme of "Little Hans".

of probability of the stochastic model of counterpoint and that of imitation. In this formulation, dynamic programming can be used to search the solution. We reported the results of experiment to generate musical pieces by the proposed method. The results showed that proposed method can generate the pieces which satisfy the requirements of counterpoint that are included in two successive beats. The results also showed that flexible imitations can be realized by the effect of the stochastic model of imitation. However, a week point of the proposed method which should be improved was revealed. The week point is that the proposing method can not prevent the occurrences of violations or anfavorable things for counterpoint which can not be included in two successive beats. This week point is considered to be caused by the approximation by a simple Markov model.

There are several future tasks;

A. Dealing with the requirements of counterpoint which can not be included in two successive beats.

B. Extensive statistical learning of the probabilities.

C. Extension of the models for more than three voices.

D. Treating note values which are smaller than eighth note.

E. Introduction of tonal harmony and modulations.

F. Modeling of the characteristics of composers or instruments.

For A, re-evaluation of the N-best solutions might resolve the problem. For B, extensive statistical learning may enable us to obtain the values of probabilities more accurately. Furthermore, it may be possible that we can generate pieces which have diverse tendencies depending on the selection of the pieces for the statistical learning. For C, adding terms which are related to all the voices might realize the models for more than three voices. For D, when we treat smaller note values than eighth note, the problem of the increase of the number of the states or the problem of lack of unity as a piece of music may occur. To avoid the occurrence of these problems, restriction for the rhythm patterns might be effective. The restriction might reduce the number of the states and help to realize the unity as a piece of music. For E, including the plan of code progressions and modulations into the structure S will be necessary to deal with the counterpoint after the Baroque period. For F, introducing the feature quantities of composers or instruments by the musical knowledge or the methods of data mining may realize finer modelings.

acknowledgment: This study was carried out partially by the support of the Ministry of Education, Culture, Sports, Science and Technology Grants-in-Aid for Scientific Research (Fundamental Research A)(Task Number 00303321) and CREST by Japan Science and Technology Agency.

7. REFERENCES

[1] J. J. Fux: *Gradus ad Parnassum*, 1725. Translation by Y. Sakamoto: *Classical Counterpoint*, Ongakunotomo-sha, 1950 (in Japanese).

[2] Y. Hasegawa: *Counterpoint*, Ongakunotomo-sha, 1955 (in Japanese).

[3] William Schottstaedt: "Automatic Counterpoint," in Max V. Mathews and John R. Pierce, editors, *Current Directions in Computer Music Reseatrch*, The MIT Press, 1989.

[4] Hibiki Yoshikawa, Mitsuru Nakai, Hiroshi Shimodaira, Shigeki Sagayama: "Automatic Counterpoint using Dynamic Programming," Proc. of the 2000 Joint Conference of Hokuriku Chapters of Electrical Sosieties, F-82, Sep., 2000 (in Japanese).

[5] Shohei Nakagata: "Automatic Counterpoint based on Dynamic Programming," Master Thesis, The University of Tokyo, 2005 (in Japanese).

[6] M. Farbood and B. Schoner: "Analysis and synthesis of Palestrina-style counterpoint using Markov chains," *Proceedings of the International Computer Music Conference*, pp. 471-474, 2001.

[7] Riyu Tamura, Yasuhiro Tajima, Yoshiyuki Kotani: "Automatic Sub-Melody Generation Using Hidden Markov Models of Pitch and Duration," IPSJ Technical Report, 2007-MUS-69, pp.7-12, 2007 (in Japanese).

[8] David Cope: *Virtual music: computer synthesis of musical style*, The MIT Press. 2000.

EXPLORING TIMBRE SPACES WITH TWO MULTIPARAMETRIC CONTROLLERS

Chris Kiefer
University of Sussex, Brighton, UK
c.kiefer@sussex.ac.uk

ABSTRACT

This paper describes the development so far of a system that uses multiparametric controllers along with an interactive high-level search process to navigate timbre spaces. Either of two previously developed interfaces are used as input devices; a hand tracking system and a malleable foam controller. Both interfaces share the property of streaming continuous multiparametric codependent data. When these data streams are mapped to synthesis parameters, the controllers can be used to explore the parameter space in an embodied manner; with the hand tracker, moving or changing the shape of the hand changes the sound, and with the foam, deforming its shape changes the sound. The controllers become too sensitive with larger parameter spaces, so a navigation system was developed to enable high level control over the subset of the parameter space in which the controllers are working. By moving and refining the working range, a timbre space can be progressively explored to find a desired sound. The search process was developed by focusing on three scenarios, the control of four, ten and forty dimensional timbre spaces. The system is used bimanually, while one hand is used for detailed search with one of the input devices, the other hand controls high level search parameters with MIDI and the computer keyboard.

Initial reactions from two musicians indicate the development so far to be successful, the next stage in this project is to carry out formal user studies.

1. INTRODUCTION

This paper explores the application of two multiparametric controllers as tools for timbre space navigation, and the development of high-level search strategies to complement their use in this context.

A previous study with an EchoFoam controller [1], a malleable foam interface, showed its potential for exploring timbre spaces. During the study, the controller was set up to navigate preset fixed subsets of a six-dimensional phase modulation synthesis patch. The participants manipulated the foam controller in order to change the synthesis parameters, in effect exploring the sound haptically . The results suggested it would be interesting to test high-level strategies for controlling the subset of parameter space within which the controller was working, in order to help the user explore the full space more effectively. The results from the study also showed some participants perceiving the controller as producing slightly unpredictable or imprecise output, so imposing a larger controllable constraint onto its output could complement it well, making an interesting combination of order and unpredictability.

Another controller developed by the author, Phalanger [2], shares similar low level properties to the foam controller of continuously streaming codependent multiparametric control data. Phalanger is a computer vision based hand tracking system, that outputs hand geometry data for musical control. It was decided to include both controllers in the development process, with the aim of creating a timbre space navigation engine that would accept input from either controller, opening up some interesting opportunities for comparing these two opposing styles of tangible and intangible interaction.

The project was approached with the following questions in mind:

1. Is it possible to navigate subsets of a timbre space and gradually zoom in to the sound you are searching for?

2. How do these two controllers and this approach compare to conventional editing tools such as a GUI or knobs and sliders?

3. What are the differences between the two types of controller in this context?

4. Will this process plug in to any synthesis process? i.e. is it possible to explore any continually controllable synthesis algorithm without knowledge of its underlying workings?

This paper describes the progress so far on this project, which has been developed to the point where the first formal evaluation is about to start. The development process that addressed the above questions will be described, but first the motivations behind the project are explored.

2. MOTIVATION

Djajadiningrat et. al [3] take the view that the body is typically neglected in interaction:

> ‘Current interfaces indeed seem to be built on the assumption that interaction can be captured

> in schemata and that the body is merely a mechanical executor. This view, however, does not do justice to our embodiment in the world.'

Extending this theme to the use of the hand for creativity in interaction, Treadaway [4] observes that current digital technology is 'deficient in utilising bodily intelligence'.

Exploring the use of bodily intelligence for controlling digital music tools is the main motivation for this project. The two controllers used here take an embodied approach to musical control compared to conventional interfaces such as the mouse and GUI or knobs and sliders. These conventional interfaces are typically up front, presenting their function to the user and enabling direct control of parameters. In contrast, these two interfaces output parallel streams of codependent parameters, and the user is required to employ their perceptual-motor skills to explore how the interface and mappings will function. This use of perceptual-motor skills is at the core of playing acoustic music where an embodied relationship with an instrument is fundamental to creating music with it. Conventional digital music interfaces in contrast can lack this embodied interaction, having more akin to scientific tools.

Further to this, another motivation is to explore accuracy and unpredictability of control. Gelineck and Serafin [5] commented on this issue in their survey of electronic musicians. 16 of 18 of the musicians they interviewed said they preferred tools that they don't fully understand or are unpredictable in some way. The two interfaces used in this project have accuracy proportional to the skill of the user and were perceived in previous studies by some users as unpredictable and imprecise compared to conventional controllers. It's compelling to see how this style of interaction can fit into the more precise world of digital music by imposing a structured framework on them. By applying these two controllers to the task of timbre space navigation, this issue can be explored further.

3. RELATED WORK

In [3], Djajadiningrat et. al. explore the use of multiparametric interfaces, suggesting these interfaces increase the bandwidth of interaction, allowing them to exploit motor skills for more sophisticated control. Focusing on music, Wanderley and Depalle [6] propose that control with multiple continuous parameters provides a more musical way of interacting with computers, moving towards the goal of control subtlety similar to acoustic instruments. Rovan et. al. [7] classify mapping schemes as one-to-one, divergent or convergent, suggesting that convergent mappings have higher potential for expressivity. Hunt and Kirk [8] carried out user studies with a multiparametric interface composed of a mouse and sliders, concluding that this class of interface can be beneficial to musicians, and also that mappings which are not one-to-one are more engaging.

In the area of timbre space navigation, evolutionary methods (e.g. [9]) have been well researched. Using interactive evolution, synthesis patches are encoded as individuals in a larger population and evolved with genetic techniques to navigate the search space. Although effective, these techniques suffer from the bottleneck of human evaluation of fitness, which reduces the efficiency of the process. Seago et. al. [10] explored timbre space navigation, proposing two strategies, multidimensional line search and the use of adapted bayesian filters. A controller-based approach is taken by van Nort and Wanderley [11], who used a graphics tablet and mapping engine to navigate complex sonic spaces.

This project fits between these areas, applying multiparametric controllers to the problem of timbre space navigation. The development of the system will now be described, starting by introducing the controllers.

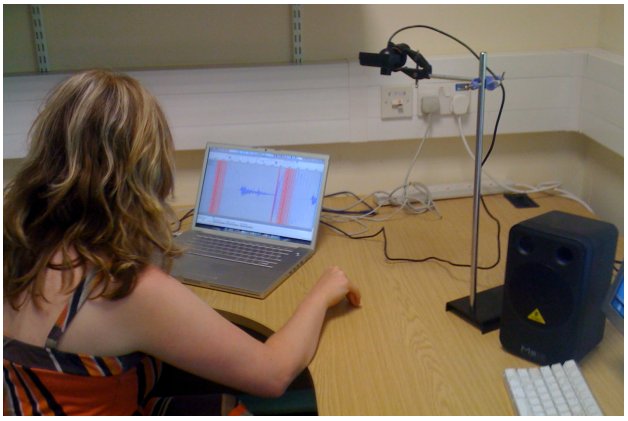


Figure 1. Phalanger in use

4. TWO MULTIPARAMETRIC CONTROLLERS

These controllers afford two styles of interaction: tangible, through the manipulation of malleable material, and intangible, through computer vision based hand tracking. Although they differ in these terms, from the point of view of the data they output they are very similar; both systems output a continuous multiparametric stream of data, in which, due to the physical nature of the devices, the parameters are codependent. Both of these system were developed using the OpenFrameworks [1] C++ toolkit on Mac OS X.

4.1 A Hand Tracking Interface

The Phalanger interface is a computer vision based hand tracking system. It's a markerless system, so no wearables are required to use it, instead neural network based skin colour tracking is used to differentiate the hand from the background. Having separated out the hand, algorithms from the openCV [2] library are employed to analyse the geometry of the hand shape; these parameters (listed in table 1) can be streamed to other applications for continuous control. Another layer of the system tracks hand shape using Support Vector Machines, however this is not used in this project; instead the streams of hand geometry data are used for timbre space navigation. For hardware, the system uses a low-cost Sony PS3Eye USB camera at 320x240

[1] http://www.openframeworks.cc/

[2] http://sourceforge.net/projects/opencvlibrary/

1	Coordinates of the topmost point
2	Coordinates of the leftmost point
3	Coordinates of the rightmost point
4	Coordinates of the centroid of the hand
5	The angle of the hand
6	The area of the hand

Table 1. Hand geometry data

resolution. The camera is mounted on a retort stand; in this case the camera is positioned to point down at the hand on a desktop.

4.2 Malleable Foam

The EchoFoam controller, an increment to the design previously described in [1], is a malleable foam controller designed with the aim of enabling nuanced and intuitive musical control. The device is constructed from conductive foam, and exploits the property that this material changes electrical resistance when deformed, in order to track the shape of the foam. Rather than using separate sensors, the controller is made from one continuous piece of foam, with embedded contact wires measuring the resistances between different points. The controller is constructed from foam squares that are glued together into a cube. Contact wires are placed in the top and bottom squares, in the four corners and in the centre. These two sets of wires are connected to two separate 74HC4051 (de)multiplexer chips on a circuit board controller by an Arduino [3] . The two sets are divided into live and sensor wires; a program on the Arduino sequences the (de)multiplexers so that a single wire is made live on one side of the foam while the five wires on the other side are scanned to measure resistance between these contacts and the live point. In this way, ten contacts can be used to take twenty-five resistance readings from the foam. These measurements, in combination, provide a consistent signature describing the shape of the foam in any particular state of deformation. Figure 2 shows the controller in use.

This system is tightly coupled with the use of Echo State Networks (ESNs), which are used as mapping engines to create control streams from the foam sensor data.

5. MAPPING WITH ECHO STATE NETWORKS

ESNs are a class of recurrent neural network, belonging under the banner of *Reservior Computing techniques* [12, 13]. They can be trained to approximate arbitrary dynamical systems, making them very useful tools for the temporal processing of multiparametric data streams, such as the outputs from EchoFoam and Phalanger. An ESN consists of a set of input and output nodes connected to a reservoir of interconnected nodes. Figure 3 shows a simplified example of ESN topology, in practice the central reservoir would be much larger. To train an ESN, a training set is created of inputs and outputs which demonstrate the desired behaviour of the trained system. During training, the

[3] `http://arduino.cc`

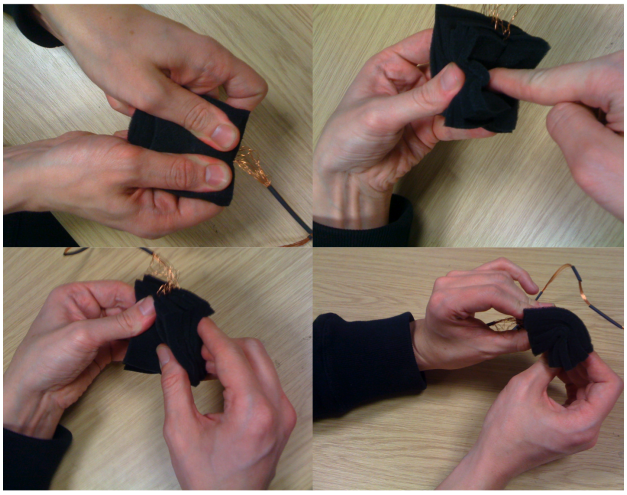


Figure 2. The Malleable Foam Controller In Use

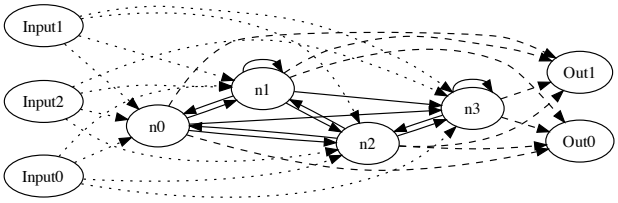


Figure 3. An Example ESN

output weights are adjusted to exploit the dynamics of the reservoir and achieve the desired behaviour. Because only the outputs weights are adjusted, training is a linear problem which is fast to solve and makes ESNs convenient and reliable to use.

In this project, ESNs are used for dimensionality reduction and as a form of mapping engine. Given the interdependent nature of the output of the foam controller, ESN mapping is a fundamental part of this system; the data from Phalanger differs because the parameters can be considered independently, however for consistency and also for the purposes of dimensionality reduction, ESNs were used to map this data as well. For each controller, their output streams are passed through an ESN and reduced in number to match the number of parameters of the synthesis engine being controller. The training process to achieve this results in an arbitrary mapping, but the outputs change consistently for each potential set of inputs to the ESN, making them reliable for musical control. Figure 4 shows an example of the inputs and outputs of an ESN being used for dimensionality reduction.

6. SYSTEM ARCHITECTURE

The timbre navigation system consisted of a collection of different programs and modules. The programs that run with each controller exist as separate OpenFrameworks applications, so an overlay application was designed that could host the timbre search engine and be integrated with both pieces of software. Figure 5 shows an overview of how the system was setup. Ableton Live was chosen as the

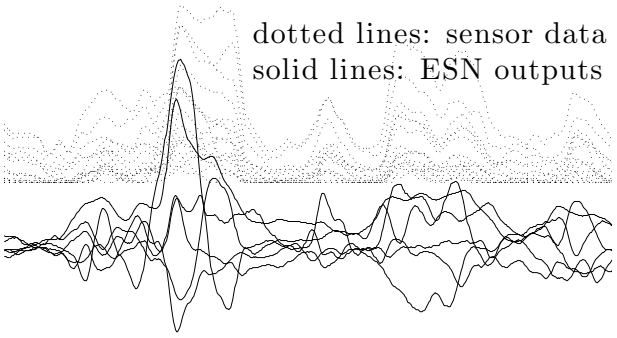


Figure 4. Echo State Network Mappings

sound engine; the software communicated with Live using liveOSC [4], an OSC/Python interface that runs as part of Live. This provides a convenient way for the timbre navigation system to query the amount, type and ranges of synthesis parameters in Live, and automatically assign control streams to them.

7. EXPLORING TIMBRE NAVIGATION STRATEGIES

What we have already is the input from two different controllers, both of which can control an arbitrary number of parameters in an arbitrary way. With the foam, the user explores the parameter space by deforming its shape, and with the hand tracker the user can manipulate parameters by moving and changing the shape of their hand. In both cases, the sound will change in direct relation to body movement. When controlling larger subsets of the parameter space, small motions amount to very large timbre changes, so in terms of controllability, working in a smaller subset is better for fine tuning of the sound. This means some kind of strategy for moving and narrowing the subset is required. Both controllers also output codependent parameters, which means that with direct mapping it will not always be possible to reach every available timbre, so a strategy is required to compensate for this.

A timbre navigation engine was designed with these issues in mind, the development of which is now described. The development took place over three progressively more complex scenarios. Overall, a *blackbox* approach was used; the system was designed with the aim of controlling any synthesis process with continuously controllable parameters, purely with controllers and no GUI.

7.1 Scenario 1: A Four Dimensional Timbre Space

The control streams were mapped to four parameters of a virtual analog synthesiser. For this scenario, the core of the navigation process was designed. To create variable subsets within which to explore with the controllers, each controlled parameter was assigned upper and lower bounds to create a working range, and also a polarity to determine if the mapping would be inverted. Pressing a key on the keyboard randomised these ranges and polarities, creating random areas of the timbre space to explore. The next step was to facilitate control of the subsets, to allow gradually refined navigation through the space. To achieve this, firstly a global percentage multiplier was applied to each range, so that the ranges could be narrowed around their center points. A continuous MIDI controller was mapped to this. Secondly, a mechanism was introduced to move the center of the ranges; pressing a key on the keyboard caused the ranges to center on the current value of each parameter, allowing navigation through the space. With these functions, the workflow for navigating the parameter space was as follows:

1. With the ranges set at full, explore different random subsets of the timbre space until something in the area of the desired result is found
2. Center on the area of the desired sound and zoom in a little by shrinking the working ranges
3. Explore the new ranges, getting closer again to the desired sound
4. Repeat the last two steps until the final result is reached

One issue was that using uniform randomness to determine the ranges sometimes led to tiny ranges on one parameter, limiting the scope of exploration. To remedy this, the random number generator was mapped through a sigmoid curve, making larger ranges more probable.

Figure 6 shows an example of how a four dimensional timbre space might be navigated. Each column represents a synthesis parameter, the shaded boxes represent the working range for that parameter and the + or - indicates mapping polarity. In (a) and (b), different random ranges are trialed to reach a suitable setting to start refining from. In (c) the ranges are re-centered and in (d) they are scaled

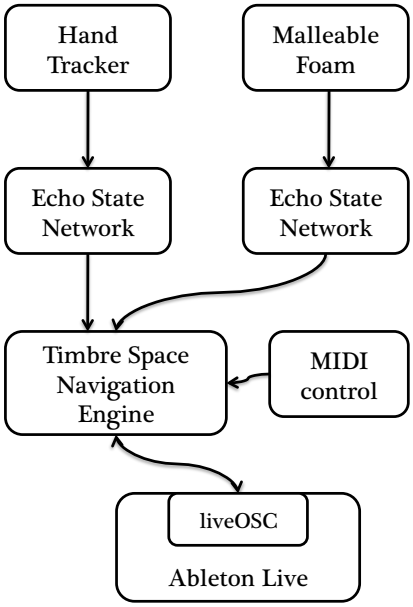


Figure 5. System Architecture

[4] http://livecontrol.q3f.org/ableton-liveapi/liveosc/

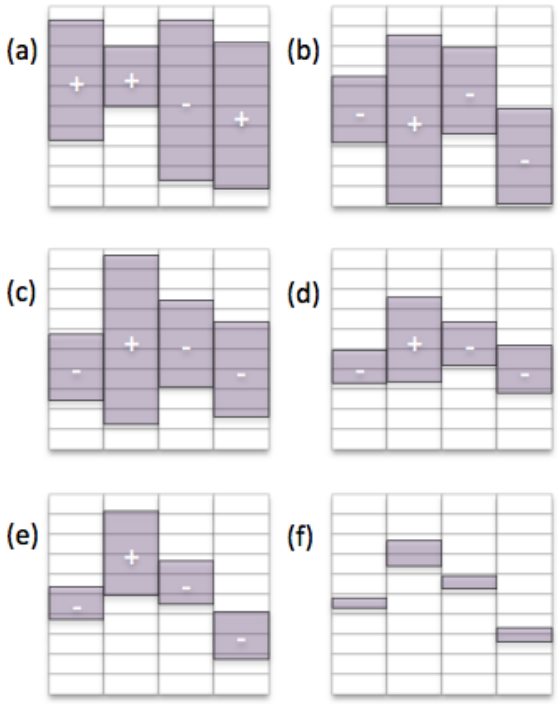


Figure 6. Exploring A Timbre Space

down. In (e) and (f), centering and scaling is repeated, arriving at a small subset of the possible space.

Overall, this solution worked satisfactorily for navigating this smaller timbre space, so the next challenge was to try exploring more dimensions.

7.2 Scenario 2: A Ten Dimensional Timbre Space

The system was mapped to ten parameters across two plugins, a sampler and a chorus that processed the sampler's output. With this amount of parameters, the process still seemed to work effectively. Up to this point, each output from the controller was mapped to the same parameter on the synthesiser. However, given the nonlinear nature on the controllers, each output stream behaves differently from another so this may limit the extent of the search space which can be navigated. To solve this, when the ranges were randomised, the parameter targets for each output stream were now randomised as well. To enable further control for the navigation process, code was added to enable mutation; pressing a key on the keyboard caused gaussian randomness between -10% and 10% to be added to the bounds of the ranges, allowed subtle variations in the exploration space. The addition of these new features helped further improve the control over navigation. The next scenario to explore was the case of when a synth had more parameters than could be reasonably output from the controllers.

7.3 Scenario 3: A Forty Dimensional Timbre Space, Navigated With 10 Control Streams

The VOPM [5] softsynth was chosen for this scenario, an FM synthesiser with over forty parameters; forty of these were selected for control by the navigation engine. This was an interesting challenge as it involved nonlinear control of a highly nonlinear soundspace, and also because FM suffers from difficultly in mapping between gestural input and synthesis parameters [14]. At first this scenario seemed to produce no sound on many settings, it was found that this silence was caused by one key parameter when the value was above 20%, so this parameter was removed from the set of targets.

As there were less control streams than parameters to control, the target parameters were selected as a random subset of the available targets, and could be changed again randomly by pressing a key on the keyboard. Selecting a new random set of targets left the previous targets on the values they were at when the settings changed, so each new random jump navigated further through the timbre space. The nonlinear nature of this timbre space meant that some settings were silent or would jump very suddenly to a different sound, so a one level undo function was added enabling the user to jump back from an unwanted setting. Navigating through a larger set of target parameters with random target selection in this manner allowed each new setting to be explored in an embodied way with the controllers, and rejected with the undo function if the new set of targets didn't take the user in the right direction. To widen the search options further, a function was added to randomise the ranges of the currently selected targets while preserving the unselected ones.

8. INITIAL REACTIONS

The system is yet to undergo formal user evaluation, however some initial thoughts about the system were gathered from two musicians. During both interviews, debugging information was showing on the screen at the start; both musicians found the experience to be much improved with the visuals removed so they could concentrate on the controller. Most importantly, both musicians found the system engaging to use. Both preferred the hand tracker as the controller, finding it easy to keep points of reference. One attempted to navigate from a distorted sound to a clean sound and back again, and achieved this successfully. They found that with the hand tracker they could return consistently to a previous point in the timbre space, and felt that their ability to do this would improve with practice.

One issue was with using the range centering process; when the working ranges are moved to a new centre, the position on the controller now corresponds to a new position in the parameter space so the sound changes. This can disrupt the flow of navigating the search space, as it's sometimes difficult to find where the sound you had centered on is in the new search space, although it should always be possible to find it. Another issue was with the size

[5] http://www.geocities.jp/sam_kb/VOPM/

of ranges, one interviewee felt it would be better to always start from much wider ranges or a completely full range.

9. DISCUSSION

The initial reactions demonstrate that the system can work successfully, although some refinement is needed. The main issue is the interaction between the range centering process and the current state of the controller, strategies need to be found to smooth out this process which in turn will improve the flow of the navigation experience. Another issue was widening the random range selection, this could be solved by providing MIDI control of the sigmoid curve which maps the random range values; the user could determine how likely ranges were to be large.

One interviewee commented that the system was good for making broader adjustments to the sound, but really fine adjustment was difficult; bearing this in mind it's interesting to consider where this type of system fits into the composition and editing process. Gelineck and Serafin [5] observe that musicians need more accurate control when they come to the final stages of a composition, so this system would fit in best at the earlier stages of creative exploration. Any settings discovered with the system can be fine tuned with a mouse and GUI later.

In terms of control, an interesting property of the system is the use of randomness; random values are used to move around the search space in search of a good place to begin fine-tuning parameters. This is necessitated by the blackbox approach to parameter control and also by the nature of the controllers. To determine mappings by something other than randomness, for example manual control, would require the attachment of meaning to variables in the system in relation to the sound being controlled, however given the embodied nature of the controllers, meaning in this system is derived from listening and physical interaction in an explorative process. Considering this, using randomness seems to be the most appropriate approach, although varying the type of randomness, for example with sigmoid mapping, can increase the level of control.

An interesting aspect of this system is the role of bimanual hand use. Treadaway [4], discussing hand use in creative practice, describes how in manual activities the dominant hand is used for micrometric and internally driven actions while the non-dominant hand is used for macrometric and externally driven actions, reflecting differences between the left and right brain hemispheres. This pattern of hand use is echoed with this system, one hand being employed for detailed exploration of the sound space while the other controls meta-level search parameters.

Returning to the questions posed in the introduction, on the issue of whether this system can plug in to any continually controllable synthesis process, this would seem possible but with one reservation that the parameters may need careful selection. In any synthesis engine, as observed in the FM synthesis scenario, there are certain parameters (for example master volume) that hold significance over others and should be excluded in a timbre space search. Parameter selection also depends on the intended sequencing of the sound, for example including envelope attack in the search space for a sound played as staccato would not be relevant. Setting the initial target parameters is something that could be controlled by a GUI, and is part of the wider creative search process.

Two questions in the introduction concerned the comparison of the two controllers used with the system and also the comparison of the system with conventional control methods; there is not enough data to answer these yet, and a formal user evaluation will help to find some answers.

10. CONCLUSION

A system has been presented for the exploration of timbre spaces, that uses multiparametric controllers and an interactive search process to provide both low level and high level control over navigation. The search process uses a combination of techniques to progressively move and refine a subset of the full timbre space being explored until a desired setting is found. Initial reactions to the system have been encouraging, but more data is needed to evaluate the efficacy of the system. The next step in the project is a formal user study which will help to answer the questions posed earlier in the paper, and to improve the design of the system.

11. REFERENCES

[1] C. Kiefer, "A malleable interface for sonic exploration," in *Proceedings of New Interfaces for Musical Expression*, 2010.

[2] C. Kiefer, N. Collins, and G. Fitzpatrick, "Phalanger: Controlling music software with hand movement using a computer vision and machine learning approach," in *New Interfaces For Musical Expression*, 2009.

[3] T. Djajadiningrat, B. Matthews, and M. Stienstra, "Easy doesn't do it: skill and expression in tangible aesthetics," *Personal Ubiquitous Comput.*, vol. 11, no. 8, pp. 657–676, 2007.

[4] C. P. Treadaway, "Hand e-craft: an investigation into hand use in digital creative practice," in *C and C '09: Proceeding of the seventh ACM conference on Creativity and cognition*, (New York, NY, USA), pp. 185–194, ACM, 2009.

[5] S. Gelineck and S. Serafin, "From idea to realization - understanding the compositional processes of electronic musicians," in *Audio Mostly*, 2009.

[6] M. M. Wanderley and P. Depalle, "Gestural control of sound synthesis," in *Proceedings of the IEEE*, vol. 92, pp. 632–644, 2004.

[7] J. B. Rovan, M. M. Wanderley, S. Dubnov, and P. Depalle, "Instrumental gestural mapping strategies as expressivity determinants in computer music performance," in *Kansei - The Technology of Emotion Workshop*, 1997.

[8] A. Hunt and R. Kirk, "Mapping strategies for musical performance," in *Trends in Gestural Control of Music* (M. Wanderley and M. Battier, eds.), Ircam - Centre Pompidou, 2000.

[9] P. Dahlstedt, "Evolution in creative sound design," in *Evolutionary Computer Music*, Springer London, 2007.

[10] A. Seago, S. Holland, and P. Mulholland, "Timbre space as synthesis space: towards a navigation based approach to timbre specification," in *Spring Conference of the Institute of Acoustics*, 2008.

[11] D. Van Nort and M. Wanderley, "Control strategies for navigation of complex sonic spaces," in *NIME '07: Proceedings of the 7th international conference on New interfaces for musical expression*, (New York, NY, USA), pp. 379–382, ACM, 2007.

[12] D. Verstraeten, *Reservoir Computing: computation with dynamical systems*. PhD thesis, Ghent University, 2009.

[13] M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, 2009.

[14] V. Lazzarini, J. Timoney, and T. Lysaght, "The generation of natural-synthetic spectra by means of adaptive frequency modulation," *Computer Music Journal*, vol. 32, no. 2, 2008.

DEPENDENT VECTOR TYPES FOR MULTIRATE FAUST

Pierre Jouvelot
CRI, Mathématiques et systèmes, MINES ParisTech
pierre.jouvelot@mines-paristech.fr

Yann Orlarey
Grame
orlarey@grame.fr

ABSTRACT

Faust is a functional programming language dedicated to the specification of executable monorate synchronous musical applications. To extend Faust capabilities to domains such as spectral processing, we introduce here a multirate extension of the core Faust language. The key idea is to link rate changes to data structure manipulation operations: creating a vector-valued output signal divides the rate of input signals by the vector size, while serializing vectors multiplies rates accordingly. This interplay between vectors and rates is made possible in the language static semantics by the introduction of dependent types. We present a typing semantics, a denotational semantics and a correctness theorem that show that this extension preserves the language synchonous characteristics. This new design is under implementation in the Faust compiler.

1. INTRODUCTION

Since Music III, the first language for digital audio synthesis, developed by Max Mathews in 1959 at Bell Labs, to Max [1], and from MUSICOMP, considered one of the very first music composition languages, developed by Lejaren Hiller and Robert Baker in 1963, to OpenMusic [2] and Elody [3], research in music programming languages has been very active and innovative. With the convergence of digital arts, such languages, and in particular visual programming languages like Max, have gained an even larger audience, well outside the computer music community.

Within this context, the Faust language [4] introduces a dual programming paradigm, based on a highly abstract, purely functional approach to signal processing while offering a high level of performance. Faust semantics is based on a clean and sound framework that enables mathematical correction proofs of Faust applications to be performed, while being complementary to current audio languages by providing a viable alternative to C/C++ for the development of efficient signal processing libraries, audio plug-ins or standalone applications.

The definition of the Faust programming language uses a two-tiered approach: (1) a core language provides constructs to manage signal transformations and (2) a macro language is used on top of this kernel to build and manipulate signal processing patterns. The macro language has rather straightforward syntax and semantics, since it is a syntactic variant of the untyped lambda-calculus with a call-by-name semantics (see [5]). On the other hand, core Faust is more unusual, since, in accordance with its musical application domain, it is based on the notion of "signal processors" (see below).

The original definition of Faust provided in [6] is based on monorate signal processors; this is a serious limitation when specifying spectral-based sound manipulation algorithms (such as FFT) or extending the language applicability outside the music domain, for instance for image analysis and manipulation (such as data compression). We propose here a multirate extension of Faust based on a key innovative principle: data rate changes are intertwined with vector data structure manipulation operations, i.e., creating an output signal where samples are vectors divides the rate of input signals by the vector size, while serializing vectors multiplies rates accordingly. Since Faust current definition does not offer first-class vectors, this proposal kills two birds with one stone by adding both multirate processing and vector data structures; this interplay between vectors and rates is made possible in the typing semantics of Faust by the introduction of dependent types.

The contributions of this paper are as follows: (1) the specification of a new extension of Faust for vector processing and multirate applications, (2) a static typing semantics of Faust, based on dependent types, (3) a denotational semantics of Faust (the one presented in [6] is operational) and (4) a Frequency Correctness theorem that validates the multirate synchronous nature of this vector extension.

After this introduction, Section 2 provides a brief informal survey of Faust basic operations. Section 3 is a proposal for a multirate extension of this core, which we illustrate with a simple vector application implementing a Haar-like subsampling operation. Section 4 defines the static domains used to define Faust static typing semantics (Section 5). Section 6 defines the semantic domains and rules used in the Faust dynamic denotational semantics; showing that this multirate extension of Faust indeed behaves properly, i.e, that signals of different frequencies merge gracefully in a multirate program, is the subject of the Frequency Correctness theorem. The last section concludes.

2. OVERVIEW OF FAUST

A Faust program does not describe a sound or a group of sounds, but a kind of *signal processor*, something that gets input signals, itself a function from time ticks t to values,

and produces output signals. The program source is organized as a set of definitions mapping identifiers to expressions; the keyword identifier `process` is the equivalent of `main` in C. Running a Faust program amounts to plugging the I/O signals implicity used by `process` to the actual sound environment, such as a microphone and an audio system, for instance.

To begin with, here are two very simple Faust examples. The first one produces silence, i.e., a signal providing an infinite supply of 0s:

```
process = 0;
```

Note that `0` is an unusual signal processor, since it takes an empty set of input signals and generates a signal of constant values, namely the integer 0. The second simple example is the conversion of a two-channel stereo signal into a one-channel mono signal using the + primitive that adds its two input signals together to yield a single, summed signal:

```
process = +;
```

Faust primitives are assembled via a set of high-level composition operations, generalizations of the mathematical function composition operator ∘. For instance, connecting the output of + to the input of `abs` in order to compute the absolute value of the summed output signal can be specified using the sequential composition operator '`:`' (colon):

```
process = + : abs;
```

Here is an example of parallel composition (a stereo cable) using the operator '`,`' that puts in parallel its left and right expressions. It uses the _ (underscore) primitive that denotes the identity function on signals, akin to a simple audio cable for a sound engineer:

```
process = _,_;
```

These operators can be arbitrarily combined. For example, to multiply the input signal by 0.5, one can write:

```
process = _,0.5 : *;
```

Taking advantage of some syntactic sugar the details of which are not addressed here, the above example can be rewritten, using what functional programmmers know as curryfication:

```
process = *(0.5);
```

The recursive composition operator '`~`' can be used to create processors with delayed cycles. Here is the example of an integrator:

```
process = + ~ _;
```

where the ~ operator connects here in a feedback loop the output of + to the input of _, via an implicit connection to the `mem` signal processor which implements a 1-sample delay, and the output of _ is then used as one of the inputs of +. As a whole, `process` thus takes a single input signal s and computes an output signal s' such that $s'(t) = s(t) + s'(t-1)$, thus performing a numerical integration operation

To illustrate the use of this recursive operator and also provide a more meaningful audio example, the following 3-line Faust program defines a pseudo-noise generator:

```
random  = +(12345) ~ *(1103515245);
noise   = random,2147483647.0 : /;
process =
  (noise,vslider("noise[style:knob]",
                 0,0,100,0.1) : *),
  100 : /;
```

The definition of `random` specifies a (pseudo) random number generator that produces a signal s such that $s(t) = 12345 + 1103515245 * s(t-1)$. Indeed, the expression `+(12345)` denotes the operation of adding `12345` to a signal, and similarly for `*(1103515245)`. These two operations are recursively composed using the ~ operator, which connects in a feedback loop the output of `+(12345)` to the input of `*(1103515245)` (via an implicit 1-sample delay) and the output of `*(1103515245)` to the input of `+(12345)`.

The definition of `noise` transforms the random signal into a noise signal by scaling it between -1.0 and +1.0, while the definition of `process` adds a simple user interface to control the production of sound; the noise signal is multiplied by the value delivered by a slider to control its volume. The whole `process` expression thus does not take any input signal but outputs a signal of pseudo random numbers (see the familiar block diagram representation of this `process` in Figure 1, where the little square near the addition block denotes a 1-sample delay operator).

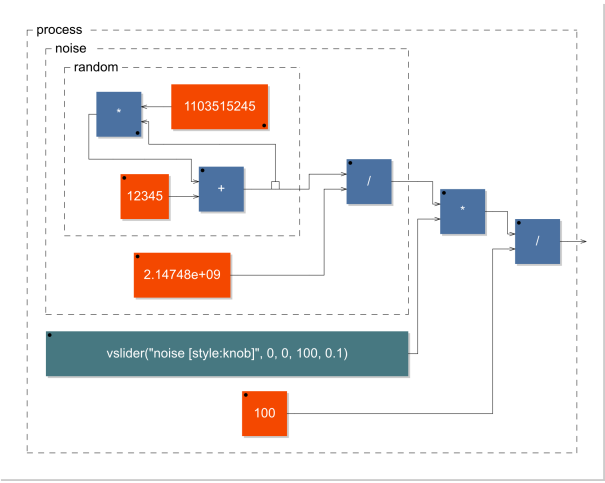


Figure 1. Noise generator `process` block diagram

The last two composition operators in the definition of core Faust, <: and :>, perform fan-out and fan-in transformations, as we illustrate in the next section

3. MULTIRATE EXTENSION

Faust, as described in [4], is a monorate language; in monorate languages, there is just one time domain involved when

accessing successive signal values. However, digital signal processing traditionally relies heavily upon subsampling and oversampling operations, which naturally lead to the introduction of multirate concepts. Since Faust targets a subset of DSP processing, the proposal introduced by Yann Orlarey [7] suggests to use multiple frequencies to deal with such issues, instead of more general clocks, such as those present in traditional synchronous programming languages [8]. We informally describe below this approach, and illustrates it with a simple example of its use.

3.1 Frequencies for vector processing

We propose to see clocking issues as an add-on to the Faust static semantics (Faust is a strongly typed language). Frequencies f are elements of the $\mathrm{Freq} = \mathbb{Q}^+$ domain. Signals, which are traditionnally typed according to the type of their codomain, will now be characterized by a pair, called a *rated type*, formed by a type and a frequency: $\mathrm{Type}^\sharp = \mathrm{Type} \times \mathrm{Freq}$.

The first key idea is to posit that multiple rates in an application are introduced via vectors. Vectors are created using the new `vectorize` primitive; informally, it collects n consecutive samples (the constant value n is provided by the signal that is the second argument to this primitive) from an input signal of frequency f and outputs vectors with n elements at frequency f/n; if the input values are of type t, then output vector samples have type $\mathrm{vector}_n(t)$. The dual `serialize` primitive maps a signal of vectors of type $\mathrm{vector}_n(t)$ at frequency f to the signal of frequency $f * n$ of their linearized elements, of type t. The primitive `[]` provides, using as inputs a signal of vectors and one of integer indexes, an output signal of successively indexed vector elements. Finally, the primitive `#` builds a signal of concatenated vectors from its two vector signal inputs.

The second key feature of this multirate extension is thus that the size n of vectors are encoded into vector types; moreover this size is provided via the *value* of a signal, argument of the `vectorize` primitive. This calls for a dependent-type [9] static semantics that embeds values within types. Since Faust strives for high run-time performance, this type system must furthermore be sophisticated enough to be able to ensure, at compile time, that a given signal is constant (when it is to be used as a signal denoting the size of a vector): we introduce intervals of values in the static semantics to deal with such an issue. Before describing formally our framework in the remainder of this paper, we illustrate it with an example.

3.2 Haar Filtering, an Example

To get a better intuitive understanding of how these vector constructs interact with Faust primitives, we present a Haar-like downsampling process, a simplified step in the Discrete Wavelet Transform shown to be of use, for instance, in some audio feature extraction algorithms [10]. The signal processor `process` takes an input signal s at frequency f and produces two output signals, the mean o_1 and difference o_2, at frequency $f/2$, such that $o_1(t) = (s(2t) + s(2t+1))/2$ and $o_2(t) = o_1(t) - s(2t+1)$. It could be defined in our extended Faust as follows:

```
down    = vectorize(2) : [](1);
mean    = _ <: _,mem :> /(2);
left    = _,!;
process =
       _ <: (mean:down),down <: left,-;
```

Here, `down` gathers the data from its input signal in pairs stored in vectors of size 2 (hence the size `2` used in the curried version of `vectorize`) from which the second element is extracted, again using a signal processor, here `[]`, curried over its second argument `1` (vector indices start at 0). This function downsamples its input signal of frequency f into an output signal of frequency $f/2$, picking one value over two from the input.

The definition of `mean` indicates that its input signal s (denoted by `_`) is duplicated, using the `<:` fan-out operator. Two copies are expected since the output of `<:` is fed into a parallel composition of two one-input signals: the first copy is simply passed along by `_`, while the second one is being delayed via `mem` by one sample. Both signals $s(t)$ and its delayed copy $s(t-1)$ are then averaged, using the fan-in operator `:>`, which adds the mixed signals to $s(t) + s(t-1)$; this sum signal is then divided by `2` using a curried division operation to yield an average signal $m(t) = (s(t) + s(t-1))/2$.

The signal processor `process` duplicates its single input s (as before, `_`) to a two-input parallel process: the first copy is averaged using `mean` and then downsampled using sequencing with `down`, yielding signal m_2; the second copy is simply downsampled, yielding s_2. These two signals are then fanned-out into the four-input signal processor `left,-`; it indeed takes four inputs, since (1) `left` takes a pair of signals, here (m_2, s_2), keeping only its left component m_2 using the primitive `!` that maps, by definition, its own s_2 to nothing and (2) the substraction operation `-` takes two inputs, here again m_2 and s_2, yielding the signal $m_2 - s_2$. The end result is the expected pair of signals $(o_1, o_2) = (m_2, m_2 - s_2)$ of downsampled means and differences.

4. STATIC DOMAINS

The multirate extension of Faust static semantics relies heavily on dependent typing, which is formally defined below.

4.1 Dependent Types

Since the values embedded in signals are typed, the static typing semantics of extended Faust uses basic types b in Base, which is a defined set of predefined types:

$$b \in \mathrm{Base} \quad = \quad \mathrm{int} \mid \mathrm{float}$$

Since our type system uses dependent types, we need a way to abstract values to yield a decidable framework. We introduce spans a in Span, which are pairs of signed integers n or m; spans represent the intervals of values that expressions may have at run time:

$$\begin{aligned} n, m \in \mathbb{Z}^\omega &= \{-\omega, +\omega\} \cup \mathbb{Z} \\ a \in \mathrm{Span} &= \mathbb{Z}^\omega \times \mathbb{Z}^\omega \end{aligned}$$

where we assume the usual extensions of arithmetic operations on $\mathbb{Z}$ to $\mathbb{Z}^\omega$; we take care in the following to avoid introducing meaningless expressions such as $-\omega{+}{+}\omega$. Note that we use integer spans here for both integer and floating-point values for simplicity purposes; extending our framework to deal with floating-point spans is straightforward. A span $a = (n, m)$ is written $[n, m]$ in the sequel.

All base-typed expressions will be typed with an element b of Base, together with a span $[n, m]$ that specifies an over-approximation of the set of values these expressions might denote. Vectors, as groups of n values, will be typed using their size (the number n) and the type of their elements. Finally, since signed integers are part of types, via spans, we will need to perform some operations over these values, and thus introduce the notion of type addition. The type domain is then [1] :

$$\begin{aligned} t \in \mathrm{Type} \quad = \quad & \mathrm{Base} \times \mathrm{Span} \mid \\ & \mathbb{N} \times \mathrm{Type} \mid \\ & \mathrm{Type} \times \mathrm{Type} \end{aligned}$$

As a short hand, we note $b[a]$ for base types, $\mathrm{vector}_n(t)$ for vector types and $t + t'$ for the addition of two types.

Not all combinations of these type-building expressions make sense. We formally define below the notion of a well-formed type:

Definition 1 (Well-Formed Type $\mathit{wff}(t)$**)**
A type t is well-formed*, noted $\mathit{wff}(t)$, iff:*

- *when $t = b[n, m]$, then $n \leq m$ and $\neg(n = m = -\omega)$ and $\neg(n = m = +\omega)$;*
- *when $t = \mathrm{vector}_n(t')$, then $\mathit{wff}(t')$ and $n \geq 0$;*
- *when $t = t' + t''$, then $\mathit{wff}(t')$ and $\mathit{wff}(t'')$.*

4.2 Rated Types

Since vectors are used to introduce multirate signal processing into Faust, we need to deal with these rate issues in the static semantics. As hinted above, we use frequencies f in Freq to manage rates:

$$f \in \mathrm{Freq} \quad = \quad \mathbb{Q}$$

In our framework, the only signal processing operations that impact frequencies are related to over- and sub-sampling conversions. To represent such conversions, we use multiplication and division arithmetic operations, thus defining Freq as the set of positive rational numbers.

The static semantics of signals manipulated in our extended Faust thus not only deals with value types, but also with frequencies. We link these two concepts in the notion [2] of *rated types* $t^\sharp$ in $\mathrm{Type}^\sharp$:

$$\begin{aligned} t^\sharp \in \mathrm{Type}^\sharp \quad = \quad & \mathrm{Type} \times \mathrm{Freq} \mid \\ & \mathrm{Type}^\sharp \times \mathrm{Type}^\sharp \end{aligned}$$

We will note t^f the rated type (t, f) and $t^\sharp {+} t'^\sharp$ the addition of two rated types. We also use simply t when f is not needed and there is no risk of confusion.

[1] The use of the same symbol, t, for both times and types should not be confusing, since they operate in different semantics.

[2] The notation $\sharp$ is, of course, different from the vector concatenation Faust primitive.

4.3 Impedances

A Faust signal processor maps sets (we called these *beams*) of signals to beams of signals. These beams have a type (we only represent the type of the image of a signal, since the domain is always time, and signals can only embed values of a single type) called an *impedance* z in Z. Type checking a Faust expression amounts to verifying the compatibility of the input and output impedances of its composed subexpressions:

$$z \in \mathrm{Z} \quad = \quad \bigcup_{n \geq 0} \mathrm{Type}^{\sharp^n}$$

The null impedance, in $\mathrm{Type}^{\sharp^0}$, is $()$, and is used when no signal is present. A simple impedance is (t^f), and is the type of a beam containing one signal that maps time to values of type t at frequency f. The impedance length $|z|$ is defined such that $z \in \mathrm{Type}^{\sharp^{|z|}}$. The i-th rated type in z $(1 \leq i \leq |z|)$ is noted $z[i]$. Two impedances z_1 and z_2 can be concatenated as $z = z_1 \| z_2$, to yield an impedance in $\mathrm{Type}^{\sharp^{d_1+d_2}}$ where $d_i = |z_i|$, defined as follows:

$$\begin{aligned} z[i] \quad &= \quad z_1[i] \ (1 \leq i \leq d_1) \\ z[i + d_1] \quad &= \quad z_2[i] \ (1 \leq i \leq d_2) \end{aligned}$$

To build more complex impedances, we introduce the $\|$ iterator as follows:

$$\begin{aligned} \|_{n,n',d} M \quad = \quad & (), \text{ if } n > n' \\ & M(n) \parallel \|_{n+d,n',d} M \text{ otherwise} \end{aligned}$$

where M is a function that maps integers to impedances. Intuitively, $\|_{n,n',d} M$ is the concatenation of $M(n), M(n+d), M(n+2d), ..., M(n')$. As a short hand, $z[n, n', d]$, which selects from z the types from the n-th type to the n'-th one by step of d , is $\|_{n,n',d} \lambda i.z[i]$, while a simple slice of z is $z[n, n'] = z[n, n', 1]$.

Definition 2 (Well-Formed Impedance $\mathit{wff}(z)$**)**
An impedance z is well-formed*, noted $\mathit{wff}(z)$, iff, for all $i \in [1, |z|]$, there exist f_i, noted $\sharp(z[i])$, and t_i such that $z[i] = {t_i}^{f_i}$, with $\mathit{wff}(t_i)$ and $f_i \in \mathrm{Freq}$.*

4.4 Schemes

Some Faust processors, such as the identity processor _ or the delay processor `mem`, are polymorphic. The static definitions of Faust primitives must thus be type schemes that abstract their input and output impedances over abstractable sorts S, in Sort. Type schemes k in Scheme are defined as follows:

$$\begin{aligned} S \in \mathrm{Sort} \quad &= \quad \{\mathrm{Base}, \mathbb{N}, \mathrm{Type}, \mathrm{Freq}, \mathrm{Type}^\sharp\} \\ k \in \mathrm{Scheme} \quad &= \quad (\mathrm{Var} \times \mathrm{Sort})^* \times \mathrm{Z} \times \mathrm{Z} \end{aligned}$$

For readability [3], we note $\Lambda x : S ... x' : S'.(z, z')$ the scheme $(((x, S), ..., (x', S')), z, z')$, where x are abstracting variables in Var. These schemes will be instantiated

[3] Keeping with a long tradition, we choose the usual ":" sign to denote typing relations, even though it is also used to represent the sequence operation in Faust. The reader should have no problem distinguishing both uses.

where needed; the substitution $(z, z')[l'/l]$ of a list l of variables by elements in l' in a pair (z, z') is defined as usual.

The static definitions of Faust primitives are gathered in type environments T that map Faust identifiers to schemes.

5. STATIC SEMANTICS

The static semantics specifies, by induction on Faust syntax, how impedance pairs are assigned to signal processor expressions. We first define some utilitary operations on static domains, and then provide static rules for Faust.

5.1 Syntax

Faust syntax uses identifiers `I` from the set Ide and expressions `E` in Exp. Numerical constants, be they integers or floating point numbers, are seen as predefined identifiers. The syntax of core Faust is thus defined as follows:

$$\begin{array}{rcl} \mathtt{E} & ::= & \mathtt{I} \mid \\ & & \mathtt{E}_1 : \mathtt{E}_2 \mid \mathtt{E}_1, \mathtt{E}_2 \mid \\ & & \mathtt{E}_1 <: \mathtt{E}_2 \mid \mathtt{E}_1 :> \mathtt{E}_2 \mid \\ & & \mathtt{E}_1 \sim \mathtt{E}_2 \end{array}$$

In Faust, every expression represents a signal processor, i.e., a function that maps signals, which are functions from time to values, to other signals.

5.2 Impedance Matching

Complex Faust expressions are constructed by connecting together simpler processor expressions. In the case of fan-in (respectively fan-out) expressions, such connections require that the involved signal processors match in some specific sense: Faust uses the *impedance matching* relation $z'_1 \succ z_2$ (resp. $\prec$) to ensure such compatibility conditions. Such a relation goes beyond simple type equality by authorizing a larger (resp. smaller) output z'_1 to fit into a smaller (resp. larger) input z_2, using the following definitions ($\succ$ requires mixing of signals, while $\prec$ simply dispatches the unmodified signals) in which $d'_1 = |z'_1|$ and $d_2 = |z_2|$:

$$\begin{array}{rcl} z'_1 \succ z_2 & = & d'_1 d_2 \neq 0 \text{ and} \\ & & \mathrm{mod}(d'_1, d_2) = 0 \text{ and} \\ & & \displaystyle\sum_{i \in [0, d'_1/d_2 - 1]} z_1[1 + id_2, (i+1)d_2] = z_2 \\ z'_1 \prec z_2 & = & d'_1 d_2 \neq 0 \text{ and} \\ & & \mathrm{mod}(d_2, d'_1) = 0 \text{ and} \\ & & \|_{1, d_2, d'_1} \lambda i. z'_1 = z_2 \end{array}$$

where equality on impedances is defined by structural induction and "mod" denotes the arithmetic modulo operation.

Since we deal in our framework with dependent types (values, via spans, appear in the static domains), performing the mixing of signals, as above, require the ability to perform, in the static semantics, additions over impedances and, consequently, over types; for instance, mixing a signal of type $\mathrm{int}[0, 2]$ with one of type $\mathrm{int}[3, 6]$ yields a signal of type $\mathrm{int}[3, 8]$. To formalize such operations, we assume the existence of static semantics addition rules such as:

$$\begin{array}{lrcl} \text{(b+)} & b[n, m] + b[n', m'] & = & b[n + n', m + m'] \\ \text{(v+)} & \mathrm{vector}_n(t) + \mathrm{vector}_n(t') & = & \mathrm{vector}_n(t + t') \end{array}$$

The presence of values in types also induces a natural order relationship $t \subset t'$ on Type.

5.3 Type Environments

We assume that there is an initial type environment T_0 that provides the typing definitions for the predefined signal processors. For instance, $T_0(\_) = \Lambda t^\sharp : \mathrm{Type}^\sharp.((t^\sharp), (t^\sharp))$ and $T_0(+) = \Lambda t^\sharp : \mathrm{Type}^\sharp.t'^\sharp : \mathrm{Type}^\sharp.((t^\sharp, t'^\sharp), (t^\sharp + t'^\sharp))$. As a consequence of the implicit mixing introduced by the impedance matching relation $\succ$ used in fan-in operations, signal processors for numerical operators such as + must be able to deal with any type; they are thus associated to polymorphic type schemes in the type environment. Their arguments must also have the same frequency, a constraint enforced by the use of the same $t^\sharp$ in these type schemes. A similar requirement exists for constants such as `0` (which are too predefined identifiers in T_0).

Introducing the vector extension in the static semantics simply amounts to adding, beside the empty vector `{}`, of type $\Lambda f : \mathrm{Freq}.t : \mathrm{Type}.((), (\mathrm{vector}_0(t)^f))$, four bindings in the initial environment T_0:

- $T_0(\mathtt{vectorize}) =$
  $\Lambda f : \mathrm{Freq}.f' : \mathrm{Freq}.t : \mathrm{Type}.n : \mathbb{N}.$
  $((t^f, \mathrm{int}[n, n]^{f'}), (\mathrm{vector}_n(t)^{f/n}))$;
- $T_0(\#) =$
  $\Lambda f : \mathrm{Freq}.t : \mathrm{Type}.m : \mathbb{N}.n : \mathbb{N}.$
  $((\mathrm{vector}_m(t)^f, \mathrm{vector}_n(t)^f), (\mathrm{vector}_{m+n}(t)^f))$;
- $T_0(\mathtt{[]}) =$
  $\Lambda f : \mathrm{Freq}.t : \mathrm{Type}.n : \mathbb{N}.$
  $((\mathrm{vector}_n(t)^f, \mathrm{int}[0, n-1]^f), (t^f))$;
- $T_0(\mathtt{serialize}) =$
  $\Lambda f : \mathrm{Freq}.t : \mathrm{Type}.n : \mathbb{N}.((\mathrm{vector}_n(t)^f), (t^{f*n}))$.

The dependent type system is key here. In the primitive `vectorize`, we are able to specify that the vector size has to be constant, since its type uses a span restricted to be one-valued, $[n, n]$; note that the frequency f' of this signal is also irrelevant, and can be of any value. When concatenating vectors with the # processor, the resulting vector size $m + n$ sums the sizes of the input vectors. We are also able to ensure that no out-of-bound accesses can occur in Faust, since the index signal argument fed to the `[]` signal processor is constrained, at compile time, to be between 0 and the vector size, since its span is $[0, n-1]$. Finally, notice how size information impacts signal frequencies; this is key to prove the theorem of Section 6.3.

5.4 Typing Rules

Faust is strongly and statically typed. Every expression, a signal processor, is typed by its I/O impedances:

Definition 3 (Expression Type Correctness $T \vdash \texttt{E}$)
An expression E *is* type correct *in an environment T, noted $T \vdash \texttt{E}$, if there exist z and z' such that $T \vdash \texttt{E} : (z, z')$ with wff(z) and wff(z').*

The static semantics inference rules are defined in Table 1; some are rather straightforward. Rule (i) ensures that identifiers are typable in the type environment T; type schemes can be instantiated to adapt themselves to a given typing context of Identifier I. In Rule (:), signal processors are plugged in sequence, which requires that the output impedance of $\texttt{E}_1$ is the same as $\texttt{E}_2$'s input. In Rule (,), running two signal processors in parallel requires that their input and output impedances are concatenated. In Rules (<:) and (:>), the $\prec$ and $\succ$ constraints are used to ensure that a proper matching of the output of $\texttt{E}_1$ to the input of $\texttt{E}_2$ is possible.

The most involved rule deals with loops ($\sim$). Here, the input impedance z_2 of the feedback expression $\texttt{E}_2$ is constrained to be the first $|z_2|$ types of the output impedance z'. Also, the first $|z_2'|$ elements of the input impedance of the main expression $\texttt{E}_1$ must be the same as the ouput impedance of the feedback expression $\texttt{E}_2$; these looped-back signals will not thus impact the global input impedance $z_1[|z_2'|+1, |z_1|]$. Note that the output impedance $\widehat{z'}$ is here an approximation of z'. This is introduced not for semantic reasons, but to make type checking decidable while ensuring that the dependent return type is valid independantly of the unknown bounds of the iteration space:

Definition 4 (Impedance Widening $\widehat{z}$)
The widened impedance *of z, noted $\widehat{z}$, is such that $|\widehat{z}| = |z|$ and $\forall i \in [1, |z|].\widehat{z}[i] = \widehat{z[i]}$, with:*

- $\widehat{\text{vector}_n(t)^f} = \text{vector}_n(\widehat{t})^f$;
- $\widehat{b[a]^f} = b[\widehat{a}]^f$;
- $\widehat{[n, m]} = [-\omega, +\omega]$.

Basically, all knowledge on value bounds is lost under widening.

Finally, the typical Rule ($\subset$) allows types to be extended according to the order relationship induced by spans in types and basic types.

6. DYNAMIC SEMANTICS

Since Faust sees parallelism as an implementation issue, the denotational semantics for core Faust is based on standard notions and does not introduce parallel-specific concepts such as powerdomains, while remaining synchronous.

6.1 Domains

A Faust expression denotes a signal processor; as such its semantics manipulates signals, which assign various values to time events. The dynamic semantics, in particular, uses integers n, k, d, i (in $\mathbb{N}$) and times t in $\text{Time} = \mathbb{N}$.

Signals map times to values v in Val :

$$v \in \text{Val} \quad = \quad \mathbb{N} + \mathbb{R} + \bigcup_{n \geq 0} \text{Val}^n + \{\bot\} + \{?\}$$

Since the evaluation process may be non-terminating, we posit that Val is a cpo, with bottom element $\bot$; all operations in Val are strict. The value ? denotes error values (useful to denote non-existing values such as 1/0), and thus, for any Operator o and Value v different from $\bot$, we assume $o(?, v) = ?$. For a vector $v \in \text{Val}^n$, represented by tuples of n elements, we define its size $|v|$ by $v \in \text{Val}^{|v|}$.

A signal s, which is a history denoted by a function, is a member of Signal = $\text{Time} \rightarrow \text{Val}$. We define the domain $dom(s)$ of a signal s by $dom(s) = \{t / s(t) \neq \bot\}$. The size of this domain $|dom(s)|$, called its support $\underline{s}$, is a member of $\mathbb{N} + \{\omega\}$, where ω is used to deal with infinite signals. We gather signals into beams $m = (m_1, ..., m_n)$ in $\text{Beam} = \bigcup_{n \geq 0} \text{Signal}^n$.

A signal processor p in Proc is the basic constituent of Faust programs: $p \in \text{Proc} = \text{Beam} \rightarrow \text{Beam}$. We define $dim(p) = (n, n')$ such that $p \in \text{Signal}^n \rightarrow \text{Signal}^{n'}$.

The standard semantics of a Faust expression is a function of the semantics of its free identifiers; we collect these in a state r, a member of $\text{State} = \text{Ide} \rightarrow \text{Proc}$.

6.2 Denotational Rules

We assume given an initial state r_0, which binds Faust predefined identifiers to their value, such that, for instance:

$$\begin{aligned} r_0(\_) &= \lambda(s).(s) \\ r_0(+) &= \lambda(s_1, s_2).(\lambda t.s_1(t) + s_2(t)) \\ r_0(\texttt{mem}) &= \lambda(s).(\lambda t.s(t-1) \text{ if } t \geq 1, 0 \text{ if } t = 0) \end{aligned}$$

These definitions assume that $T \vdash \texttt{0} : t$ for all types t, since this is needed for the definition of mem to make sense.

As in the static semantics, introducing the vector extension in the dynamic semantics [4] simply amounts to adding, beside the value $\lambda().(\lambda t.())$ for {}, four straightforward bindings in the initial state:

- $r_0(\texttt{vectorize}) =$
  $\lambda(s_1, s_2).(\lambda t.(s_1(nt), ..., s_1(n-1+nt)),$
  $\text{where } n = s_2(0));$
- $r_0(\#) = \lambda(s_1, s_2).(\lambda t.s_1(t) \| s_2(t));$
- $r_0(\texttt{[]}) = \lambda(s_1, s_2).(\lambda t.s_1(t)[s_2(t)]);$
- $r_0(\texttt{serialize}) =$
  $\lambda(s).(\lambda t.\ \bot, \text{ if } n = |s(0)| = 0,$
  $s(\lfloor t/n \rfloor)[\text{mod}(t, n)] \text{ otherwise}).$

To be able to properly define the semantic function E:

$$E \in \text{Exp} \rightarrow \text{State} \rightarrow \text{Beam} \rightarrow \text{Beam}$$

one needs to ensure that we operate with states that are type-correct.

Definition 5 (State Type Correctness $T \vdash r$)
A state r is type correct *in an environment T, noted $T \vdash r$, if, for all* I *in $dom(r)$, one has $T \vdash \texttt{I}$.*

[4] We consider that all notations introduced to manipulate impedances can similarly be applied to vectors and beams.

$$\text{(i)}\ \frac{T(\mathtt{I}) = \Lambda l.(z,z') \quad \forall (x,S)\in l \ . \ l'(x)\in S}{T \vdash \mathtt{I} : (z,z')[l'/l]}$$

$$\text{(:)}\ \frac{T \vdash \mathtt{E}_1 : (z_1,z_1') \quad T \vdash \mathtt{E}_2 : (z_1',z_2')}{T \vdash \mathtt{E}_1 : \mathtt{E}_2 : (z_1,z_2')}$$

$$\text{(<:)}\ \frac{T \vdash \mathtt{E}_1 : (z_1,z_1') \quad T \vdash \mathtt{E}_2 : (z_2,z_2') \quad z_1' \prec z_2}{T \vdash \mathtt{E}_1 <: \mathtt{E}_2 : (z_1,z_2')}$$

$$\text{(,)}\ \frac{T \vdash \mathtt{E}_1 : (z_1,z_1') \quad T \vdash \mathtt{E}_2 : (z_2,z_2')}{T \vdash \mathtt{E}_1,\mathtt{E}_2 : (z_1\|z_2, z_1'\|z_2')}$$

$$\text{(:>)}\ \frac{T \vdash \mathtt{E}_1 : (z_1,z_1') \quad T \vdash \mathtt{E}_2 : (z_2,z_2') \quad z_1' \succ z_2}{T \vdash \mathtt{E}_1 :> \mathtt{E}_2 : (z_1,z_2')}$$

$$\text{(}\subset\text{)}\ \frac{T \vdash \mathtt{E} : (z,z') \quad z' \subset z_1' \quad z_1 \subset z}{T \vdash \mathtt{E} : (z_1,z_1')}$$

$$\text{(}\sim\text{)}\ \frac{T \vdash \mathtt{E}_1 : (z_1,z') \quad T \vdash \mathtt{E}_2 : (z_2,z_2') \quad z_2 = z'[1,|z_2|] \quad z_2' = z_1[1,|z_2'|]}{T \vdash \mathtt{E}_1 \sim \mathtt{E}_2 : (z_1[|z_2'|+1,|z_1|], \widehat{z'})}$$

Table 1. Faust Static Semantics

$$\begin{aligned}
E[\![\mathtt{I}]\!]r &= r(\mathtt{I}) \\
E[\![\mathtt{E}_1 : \mathtt{E}_2]\!]r &= p_2 \circ p_1 \\
E[\![\mathtt{E}_1, \mathtt{E}_2]\!]r &= \lambda m.p_1(m[1,d_1])\|p_2(m[d_1+1,d_1+d_2]) \\
E[\![\mathtt{E}_1 <: \mathtt{E}_2]\!]r &= \lambda m.p_2(\|_{1,d_2,d_1'}\lambda i.p_1(m)) \\
E[\![\mathtt{E}_1 :> \mathtt{E}_2]\!]r &= \lambda m.p_2(\|_{1,d_2,1}\lambda i.\mathrm{sum}(p_1(m)[i,d_1',d_2])) \\
&\quad \text{where } \mathrm{sum}((s)) = (s) \text{ and } \mathrm{sum}((s)\|m) = r(+)((s)\|\mathrm{sum}(m)) \\
E[\![\mathtt{E}_1 \sim \mathtt{E}_2]\!]r &= \lambda m.fix(\lambda m'.p_1(p_2(@(m'[1,d_2]))\|m)) \\
&\quad \text{where } @(()) = () \text{ and } @((s)\|m) = E[\![\mathtt{mem}]\!]rs\|@(m)
\end{aligned}$$

Table 2. Faust Denotational Semantics: we note $p_i = E[\![\mathtt{E}_i]\!]r$ and $(d_i, d_i') = dim(p_i)$

The semantics $E[\![\mathtt{E}]\!]r$ of an expression E in a type-correct state r is a function that maps an input beam m to an output beam m'.

The semantics (see Table 2) of an identifier is available in the state r. The semantics of ":" is the usual composition of the subexpressions' semantics. The semantics of a parallel composition is a function that takes a beam of size at least $d_1 + d_2$ and feeds the first d_1 signals into p_1 and the subsequent d_2 into p_2; the outputs are concatenated. The fan-out construct repeatedly concatenates the outputs of p_1 to feed into the (larger) d_2 inputs of p_2. The fan-in construct performs a kind of opposite operation; all $\mathrm{mod}(i, d_2)$-th output values of p_1 are summed together to construct the i-th input value of p_2. The loop expression has the most complex semantics. Its feedback behavior is represented by a fix point construct; the output of p_2 is fed to p_1, after being concatenated to m, to yield m'; the input of p_2 is the one-slot delayed version of m'.

6.3 Frequency Correctness Theorem

In the presence of signals using different rates at run time, the consistency of their frequency assignment must be ensured. In particular, we show below that the support of signals and, more generally, beams can be bounded in a way consistent with their relative frequencies; this is the Frequency Correctness theorem. Of course, this theorem is only valid if the values denoted by a given Faust expression are consistent with its type definition, and kept as such all along execution (see the Subject Reduction theorem linking Faust static and dynamic semantics in [11]). We proceed first with the definition of this notion of runtime type correctness.

Definition 6 (Value Type Correctness $v : t$)
A value v is type correct*, noted $v : t$, iff:*

- *when $v \in \mathbb{N}$, then $t = \mathrm{int}[n,m]$ and $n \le v \le m$;*
- *when $v \in \mathbb{R}$, then $t = \mathrm{float}[n,m]$ and $n \le v \le m$;*
- *when $v \in \bigcup_n \mathrm{Val}^n$, then $t = \mathrm{vector}_n(t'), n = |v|$ and, for all $i \in [0, n-1], v[i] : t'$.*

Definition 7 (Signal Type Correctness $s : t^f$)
A signal s is type correct *w.r.t. a type t^f, noted $s : t^f$, if, for all $u \in dom(s)$, one has $s(u) : t$.*

Definition 8 (Beam Type Correctness $m : z$)
A beam m is type correct *w.r.t. an impedance z, noted $m : z$, if $|m| = |z|$ and, for all $i \in [1, |m|]$, one has $m[i] : z[i]$.*

For the evaluation process to preserve consistency, the environment T and state r, which provide the static and semantic values of predefined identifiers, must introduce consistent definitions for their domains:

Definition 9 (State Type Consistency $\vdash T, r$)
An environment T and a state r are consistent*, noted $\vdash T, r$, if, for all* `I` *in $dom(r)$, for all z, z', m, one has: if $T \vdash$ `I` $: (z, z')$ and $m : z$, then $r($`I`$)(m) : z'$ and $dim(r($`I`$)) = (|z|, |z'|)$.*

We may now proceed with the issue of frequency.

Definition 10 (Beam Boundness (m, z) ! c)
*For any $c \in \mathbb{Q}$, a beam m of impedance z is c-*bounded*, noted (m, z) ! c, if $\min_{i \in [1,|z|]}(\underline{m[i]}/\sharp(z[i])) \leq c$.*

Informally, when (m, z) ! c, then there is at least one signal i^* in m that has at most $c\sharp(z[i^*])$ elements in its domain of definition [5] . This is interesting since the supports of signals in a beam m tell us something about how many values can be computed if we use m as input of a signal processor. Thus $c\sharp(z[i^*])$ is an upper bound on the number of elements that can be used in a synchronous computation (all subsequent values are $\perp$), thus yielding some clues about the size of buffers needed to perform it.

Another way to look at c-boundness comes from c itself; being the inverse of a frequency, its unit is the second, and thus c is a time. The definition of Beam Boundness yields an upper bound on the time required to exhaust (at least one of) the signals of m, thus providing a time limit on computations that would use these as actual inputs. Even though this limit, as stated here, holds for a complete computation, it also applies when one deals with slices of the computation process, for instance when considering buffered versions of a program.

The Frequency Correctness theorem states that, given a Faust expression `E` (with no explicit `mem`, since its delaying action extends domains of definition), if the environment T and state r are consistent and `E` maps beams of impedance z to beams of impedance z', then, given a beam m that is type correct w.r.t. z and is c-bounded, then the semantics $p(m)$ of `E` will yield a c-bounded beam m' of impedance z'.

Theorem 1 (Frequency Correctness)
For all `E` *not containing* `mem`*, T, z, z', c, r, m and m', if $\vdash T, r,\ m : z,\ (m, z)$! c, $T \vdash$ `E` $: (z, z')$, then $|z'| = 0 \vee (m', z')$! c, where $m' = p(m) : z'$ and $p = E[\![$`E`$]\!]r$.*

Basically, this theorem (see proof in [11]) tells us that an upper-bound of the running time of `E` is always the same, whichever way we try to assess it via any of its observable facets (namely input or output data): c is consistent and thus a characteristics of `E`. This shows that the synchronous nature of Faust is preserved.

[5] When signals are properly synchronized, e.g., in an actual computation, all $\underline{m[i]}/\sharp(z[i])$ are equal, and the comments in this section about i^* apply in fact to all signals.

7. CONCLUSION

We provide the typing semantics, denotational semantics and correctness theorem for a new multirate extension of Faust, a functional programming language dedicated to musical applications. We propose to link the introduction of a vector datatype in a synchronous setting to the presence multiple signal rates. We describe a dedicated framework based on a new polymorphic dependent-type static semantics in which both vector sizes and frequencies are values, and prove a synchrony consistency theorem relating values and frequencies. This proposal is under implementation in the Faust compiler.

8. ACKNOWLEDGEMENTS

This work is partially funded by the French ANR, as part of the ASTREE Project (2008 CORD 003 01).

9. REFERENCES

[1] M. Puckette, "The patcher," in *Proceedings of the International Computer Music Conference*, ICMA, 1988.

[2] G. Assayag and C. Agon, "OpenMusic Architecture," in *Proceedings of International Computer Music Conference* (ICMA, ed.), pp. 339–340, 1996.

[3] S. Letz, Y. Orlarey, and D. Fober, "The Role of Lambda-Abstraction in Elody," in *Proceedings of the International Computer Music Conference* (ICMA, ed.), pp. 377–384, 1998.

[4] E. Gaudrain and Y. Orlarey, "A Faust Tutorial," tech. rep., GRAME, Lyon, 2003.

[5] H. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*. North-Holland Publishing, 1981.

[6] Y. Orlarey, D. Fober, and S. Letz, "Syntactical and Semantical Aspects of Faust," *Soft Computing*, vol. 8, no. 9, pp. 623–632, 2004.

[7] Y. Orlarey, "Notes sur les extensions de Faust," tech. rep., GRAME, Lyon, 2009.

[8] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone, "The Synchronous Languages Twelve Years Later," in *Proceedings of the IEEE*, vol. 91, Jan. 2003.

[9] J. van Leeuwen, ed., *Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics*. MIT Press, 1990.

[10] G. Tzanetakis, G. Essl, and P. Cook, "Audio Analysis using the Discrete Wavelet Transform," in *Proc. Conf. in Acoustics and Music Theory Appl.. WSES*, 2001.

[11] P. Jouvelot and Y. Orlarey, "Semantics for Multirate Faust," tech. rep., CRI, Mathématiques et systèmes, MINES ParisTech, Nov. 2009.

THE 'STANZA LOGO–MOTORIA': AN INTERACTIVE ENVIRONMENT FOR LEARNING AND COMMUNICATION

Antonio Camurri, Gualtiero Volpe
University of Genova
{antonio.camurri, gualtiero.volpe}@unige.it

Sergio Canazza
University of Padova
canazza@dei.unipd.it

Corrado Canepa
University of Genova
corrado@infomus.dist.unige.it

Antonio Rodà, Serena Zanolla, Gian Luca Foresti
University of Udine
{serena.zanolla, antonio.roda}@uniud.it

ABSTRACT

The *Stanza Logo-Motoria* is a multimodal interactive system for learning and communication developed by means of the EyesWeb XMI platform. It is permanently installed in a Primary School where it is used as an alternative and/or additional tool to traditional ways of teaching. The *Stanza Logo-Motoria* is used by all the pupils of the school - from the first to the fifth grade - including the children with disabilities. This paper describes the system and a presents a first assessment of the teaching activities carried out using it.

1. INTRODUCTION

Today, the European Education System consists of an extremely heterogeneous environment: there is a significant diversification in levels of learning, a high proportion of foreign children and a growing number of children with disabilities [1]. In particular, the educational process of students with disabilities is a long and complex one, which is faced in different ways by various European Union countries. The European Agency for Development in Special Needs Education affirms that the trend in this field is to implement education policies which place disabled students in mainstream schools providing different types of support to teachers in terms of additional staff, teaching materials, in-service training and technical equipment. Technology can play a particularly valuable role in promoting greater adaptability of the Education System and, on the other hand, increasing the level of cultural demand [2].

Back in 1983, H. Gardner studied the different types of intelligences developing the Multiple Intelligences Theory [3]. This theory suggests that the traditional notion of intelligence, based on I.Q. testing, is far too limited. Instead, Gardner proposes eight different intelligences to account for a broader range of human potential in children and adults. These intelligences are: linguistic, logical-mathematical, spatial, bodily-kinesthetic, musical, interpersonal, intrapersonal, naturalist. Gardner says that our schools and culture focus most of their attention on linguistic and logical-mathematical intelligence. We should also place equal attention on individuals who are gifted in the other intelligences: artists, musicians, designers, dancers, and also disabled students. Unfortunately, many children who have these gifts do not receive much reinforcement for them in school. The theory of multiple intelligences proposes a major transformation in the way our schools are organized. It suggests that teachers ought be trained to present their lessons in a wide variety of ways using music, cooperative learning, art activities, role play, multimedia, field trips, inner reflection, and much more. It is very significant to recognize and nurture all human minds and all their combinations in order to encourage interaction with the world, global growth of the person, and the achievement of the highest possible level of learning [4].

In this framework, this paper presents a multimodal interactive system, *Stanza Logo-Motoria*, offering an alternative and/or additional tool to traditional ways of teaching that often do not adapt to the individual learning ability. In real-time the system analyzes the full-body movements and gestures of the children within a sensorised environment and maps them onto real-time manipulation and processing of audiovisual content. A particular focus is placed on expressive gestures, i.e., gestures containing and conveying emotional, affective information. In this way, the *Stanza Logo-Motoria* can be exploited by teachers either to convey content by means of an alternative method or also to verify the level of knowledge in children who better express their capabilities using the visual, spatial, or bodily intelligence.

The rest of this paper is organized as follows. In Sec. 2 theoretical foundations and related works are briefly introduced. Sec. 3 describes the system architecture of *Stanza Logo-Motoria*. Sec. 4 presents the feature extraction component and the description of the features the system uses. Sec. 5 describes how the system and the features extracted are used in a concrete instance of *Stanza*: the *Resonant Memory* application. In Sec. 6 details of *Stanza* activities are presented together with the first results from system assessment. Conclusions are drawn and future plans are laid

out in Sec. 7.

2. THEORETICAL FOUNDATIONS AND RELATED WORK

The *Stanza Logo-Motoria* grounds its theoretical foundations on the Enaction theories and Embodied cognition, finding a neurophysiological basis on the discovery of the mirror neuron system. Mirror neurons constitute the neural substrate for the recognition and comprehension of actions performed by other individuals. Rizzolatti and Vozza [5] emphasize the motor aspect of cognition, claiming that learning is behind the action and that the basis of knowledge is the fact that "we do things". There are two kinds of knowledge: one is scientific and objective, the other is experiential which is our primary knowledge based on our motor system and our experiences; embodiment is the necessary condition for the development of cognitive processes. This approach [6] is grounded on multi-sensory coupling of perception and action, motor imitation and issues concerning emotions and subjectivity. Each cognitive activity is always "situated", it is inextricably associated with 'what we are doing physically' and on the structure and dynamics of the environment [7]."Learning by doing" is an important theoretical dimension also for Enactive theories of cognition; Enactive knowledge is based on motor skills (such as manipulating objects) where Enactive representations are acquired by doing [8].

In the *Stanza Logo-Motoria*, it is possible to recover the important aspect of motor knowledge and to use it to resolve situations of learning difficulties. The *Stanza Logo-Motoria* follows this approach, being an environment where the users must do things to receive a content: they have to enter the space, choose a location, listen carefully, perform activities, they have a reason for learning, ways of acting and perceiving, a significant environment. Knowledge is not imposed from above but is offered: the users must "do things" to receive it. Furthermore, the pupils learn in motion, they seek the content by physically moving within the space: ideas, thoughts, concepts, and categories are shaped by aspects of the body [6]. The *Stanza Logo-Motoria* becomes an instructional agent [2] by transferring information and instructional knowledge just as the teacher presents his/her explanation to students.

By the end of Sixties, Myron Krueger begins his experiments on interactive electronic image, immediately offering artificial environments constructed manipulating visual and audio information. He presents [9] the concept of environment linked to physical space where the observer seeks to intervene. Krueger uses spaces directly modified by the user's presence without any invasive devices. In the first experiments, using a sensitized carpet, Krueger extracts the user's location within the environment (Glowflow, Metaplay, Psychic Space) and then he directly manipulated user's gestures through video tracking. The *Stanza Logo-Motoria*, like Krueger's space, is an environment where both spatial position and user's gestures are detected by a video-camera. The coordinates of the position and the measure of open arms gesture correspond in this case to a specific audio and/or video feedback. Krueger [10] emphasizes the role of the body and considers 'responsive environments' as tools for the re-appropriation of sensory faculties sacrificed by the power of audiovisual representation. In the same way, the *Stanza Logo-Motoria* is a tool to retrieve children's attention span and concentration; the *Stanza Logo-Motoria* offers children the chance to re-experience the feeling of being good and satisfied about himself during active listening.

Another important example of the use of an interactive multimedia environment with children (and especially with disabled children) is SOUND=SPACE. In 1984 Rolf Gehlhaar developed SOUND=SPACE [11], an interactive multi-user musical environment in which visitors trigger and influence the production of sounds merely by moving about an empty space surveyed by an ultrasonic echolocation system. Since its development, this system has been displayed publicly worldwide, becoming a particularly favorite for groups with special needs. SOUND=SPACE is still being explored by visitors and participants in creative workshops for special needs groups. Further on, Gehlhaar and colleagues [12] worked on a new multi-user interactive audio visual installation: CaDaReMi. CaDaReMi addresses this problem: the user's difficulty to spatially "anchor" his/her activities; CaDaReMi answers by providing a number of visual clues designed to help the user understand "how things work", to use the "spectacle" of the installations to explain it to new users and to make the sound topologies visible. The *Stanza Logo-Motoria*, like CaDaReMi, is an environment: stimulant for users to expressively explore a wide range of different sounds; collaborative as it may be used by several persons at the same time; challenging and interesting, providing a palette of both familiar and strange sounds; visually engaging, enhancing the user's experiences and promoting their ability to locate themselves and to decode the events of others in the space at the same time; socially engaging by promoting user-user interaction, thus strengthening the sense of working together; intuitive in its functionality and use as no explanations and no special expertise are required in order to obtain a first results; learnable and master-able, being sufficiently complex so that users may, in time, enjoy the experience of "getting better" at using it and, at the same time, easy enough for beginners to quickly experience success.

Unlike the environments described above, the *Stanza Logo-Motoria* is an interactive space, permanently installed at a school, which allows the assimilation of content by learning through movement. It is a container of knowledge that can be filled with any topic from History to Math, Geography or Science. The system creates a communicative interactive environment, a place where the user, moving the body in space and through a simple arm gestures, causes the production of information that, in absence of the usual modality of communication (the word), would not be possible to convey.

The *Stanza Logo-Motoria* has been developed by using the EyesWeb XMI platform (www.eyesweb.org). It addresses the following areas in the SMC Roadmap: Interactive Multimedia Systems and Augmented Action and

Perception.

3. SYSTEM ARCHITECTURE

Fig. 1 shows the overall system architecture for the *Stanza Logo-Motoria*. It consists of three major components:

- The *input component*, receiving the video stream captured by one or more video-cameras observing the space and the information gathered by possible further environmental sensors (e.g., microphones, pressure sensors on the floor). This component is also responsible for data pre-processing (e.g., denoising, background subtraction techniques to extract the silhouettes of the users).

- The *feature extraction component*, which analyzes the input data in order to get information about (i) how the user occupies the space (e.g., where they go; how long they remain in a given area), (ii) the expressiveness of their gestures. More details on this component are provided in Sec. 4.

- The *component for real-time processing of audiovisual content*, which is responsible of the real-time control and processing of audio and video material and depends on the features extracted by the *feature extraction component*.

The components of the system architecture will be described in detail with reference to *Resonant Memory*, a specific instance of the *Stanza Logo-Motoria* .

4. FEATURE EXTRACTION

In *Stanza Logo-Motoria* a video-camera is used to capture body and arm gestures. The image stream is processed and a number of features are extracted. The focus is on so-called non-invasive approaches. Analysis is grounded on a multi-layered model for expressive gesture processing [13] which aims at extracting features characterizing both the occupation of space on the part of the users and the expressive quality of their gestures (e.g., whether a gesture is smooth or impulsive, determined or hesitant, etc.). Analysis of both space occupation and gesture is usually based on the input video stream which is pre-processed in order to extract the silhouette of the users.

Space analysis starts from the trajectory followed by each user within the space (e.g., computed as the trajectory of the center of mass of the user). It is possible to define regions within the space and identify those that are currently occupied by the users (this information may be used for example to trigger possible audiovisual content). The occupation rate for each region is also computed. This can be used for characterizing the visiting behavior of the users with respect to the space. For example, if an audiovisual content is associated with a specific region, a high occupation rate of that region may indicate a high interest of the users for a specific content.

The analysis of the gestures is based on features that are extracted at different levels, from relatively simple (or low level) ones to more complex features (high-level) describing gestures in terms of categories inspired to theories and experiments from psychology and humanities.

Low-level features include basic kinematical features (e.g., position, velocity, and acceleration of the center of mass of the silhouette), and silhouette-based features, i.e., features directly computed on the silhouette of the user. Silhouette-based features include, for example, the Motion Index (i.e., the amount of movement detected by the video-camera), the Contraction Index (an index measuring the contraction/expansion of the body computed as the ratio between the area of the silhouette and the area of the bounding rectangle), the orientation of the body (computed as the orientation of the major axis of an ellipse approximating the body).

High-level features are computed from low-level features on the basis of theories, models, and experiments from psychology, biomechanics, and humanities. For example, according to Rudolf Laban's Theory of Effort [14] a gesture can be quick (impulsive) or sustained (smooth), direct or flexible, heavy or light. Relevant sources from research in psychology include the works by Wallbott [15], De Meijer [16], and Boone and Cunningham [17]. Examples of high-level features include: the Directness Index, measuring whether a gesture is direct or flexible; the Impulsivity Index, measuring whether a gesture is quick or sustained; Fluidity, measuring whether a gesture is bounded and hesitant or unbounded.

The instance of *Stanza Logo-Motoria* described in the following - *Resonant Memory* application - uses mainly low-level features that can be extracted directly from the image stream: the overall movement of a human body (Motion Index), represented as the movement of the centre-of-mass of the body, and the open arm gesture represented as the variation in the size of the body bounding box (Contraction Index). The focus here is on low-level features because they are relatively fast and easy to calculate, and at the same time, sufficient for defining a rich set of gestures for control. Low-level features are also inspired by studies on visual perception, and are used to build computational models inspired by perception [18].

5. THE *RESONANT MEMORY* APPLICATION

Resonant Memory is an instance of the *Stanza Logo-Motoria* installed in the Primary School where it is experimentally in use. In *Resonant Memory*, the space captured by a web-camera is divided into nine areas: eight of these are peripheral areas whereas the ninth is central; in this case the zones are nine but the number may vary depending on the didactic needs. Sound or visual information corresponds to each area. The trajectory of the centre of mass is used to match a sound to a specific position in space. A child explores the resonant space in which he/she can freely move without using sensors:

- Noises, sounds, and music are associated with peripheral zones and are reproduced when the child reaches and occupies a peripheral zone;

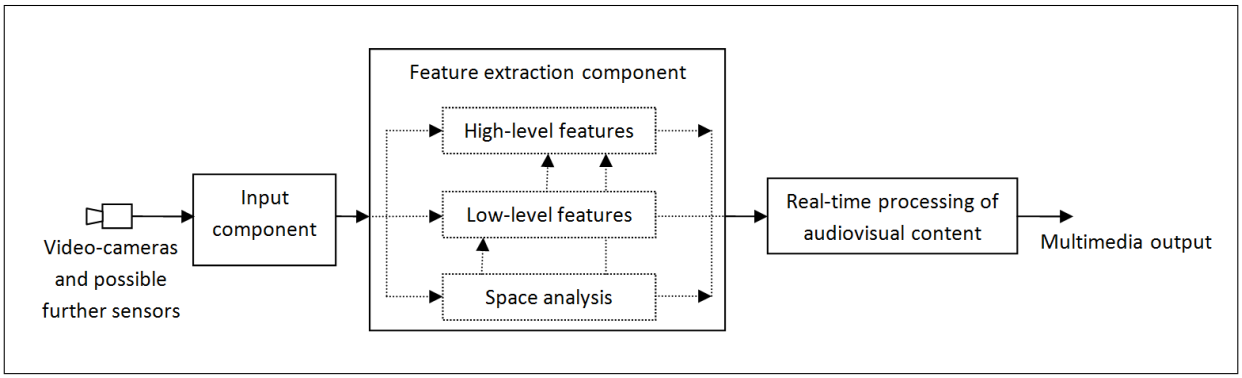


Figure 1. System architecture.

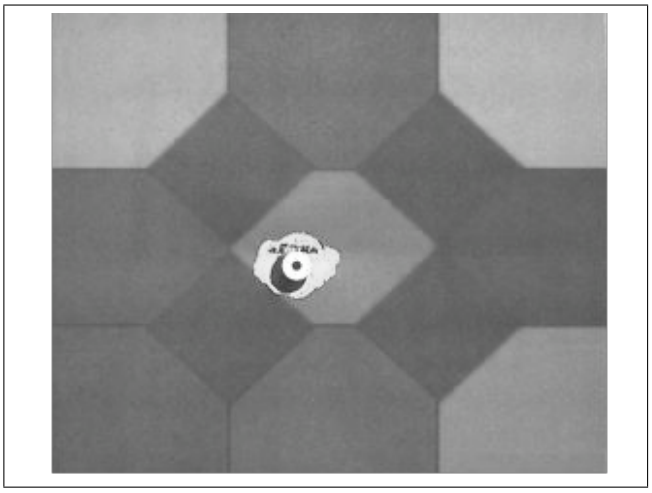


Figure 2. The sound reproduction of a story starts when the child enters the central area.

- Audio reproduction of a story is associated with the central zone (see Fig. 2); the story provides references to the various sounds located in the peripheral areas. The child, listening to the story, enjoys searching for the sounds heard before and, at the same time, he/she creates the soundtrack of the story.

The video analysis and sound rendering tasks are performed by a software patch developed in the EyesWeb environment. Fig. 3 shows an overall picture of the patch, which can be subdivided into three stages. The first is the input stage: the signal from the camera is processed in order to extract several low-level features related to the user's movements. Background subtraction is achieved via a statistical approach: the brightness/chromaticity distortion method [19]. Extracted features include the trajectory of the center of mass, the Motion Index, and the Contraction Index. In the mapping stage (see Fig. 4), the patch analyses the features and, according to the user's actions, it runs the transitions among four states: exploration, story, pause, and reset. Finally, the output stage controls the playback of a set of pre-recorded audio files.

When the application starts and the user enters in the area of activity for the first time, the application is in the exploration mode. Whenever the user reaches one of the eight peripheral zones, this information is stored by the system (see Fig. 2). If during the exploration phase the user reaches the central zone no event is triggered. Only after all the eight zones have been explored at least once and the user reaches the central area, the application turns to the story mode. When the user widens their arms during the story, the system pauses and the playback of the sound events is interrupted until the user closes their arms (see Fig. 5). If the user leaves the area of activity and does not return for over a certain time limit (which can be pre-set), the application goes back into reset state which erases the track of the visited zones. When a new user enters in the area of activity after the reset, the application starts over and he/she can begin a new exploration phase.

Currently the input modality is limited to video tracking and the only the sound modality is used as output for the following reasons:

- The path of teaching/learning by the use of *Stanza Logo-Motoria* in the school was a novelty for students and for teachers, therefore we proposed a very simple, direct and intuitive tool also considering the limited financial resources of the school.

- In agreement with teachers we thought that, in general, children need to recover the ability to listen and to experience space without any visual references. The aim is to achieve heightened awareness of space.

- Multi-modality is the ultimate goal: as children learn to control one of the modalities, the *Stanza Logo-Motoria* can evolve and several other modalities can be introduced according to their own individual paths of growth.

6. ASSESSMENT

The *Stanza Logo-Motoria* is a technological tool to enhance alternative intelligences and communication; the purpose is to promote learning motivation and develop pupils' different cognitive styles. The *Stanza Logo-Motoria* is currently installed in the "E. Frinta" Primary School in Gorizia (Italy). We are evaluating to use the *Stanza* with 10 Primary School classes, from the first to the fifth grade (a total of 170 pupils) as follows:

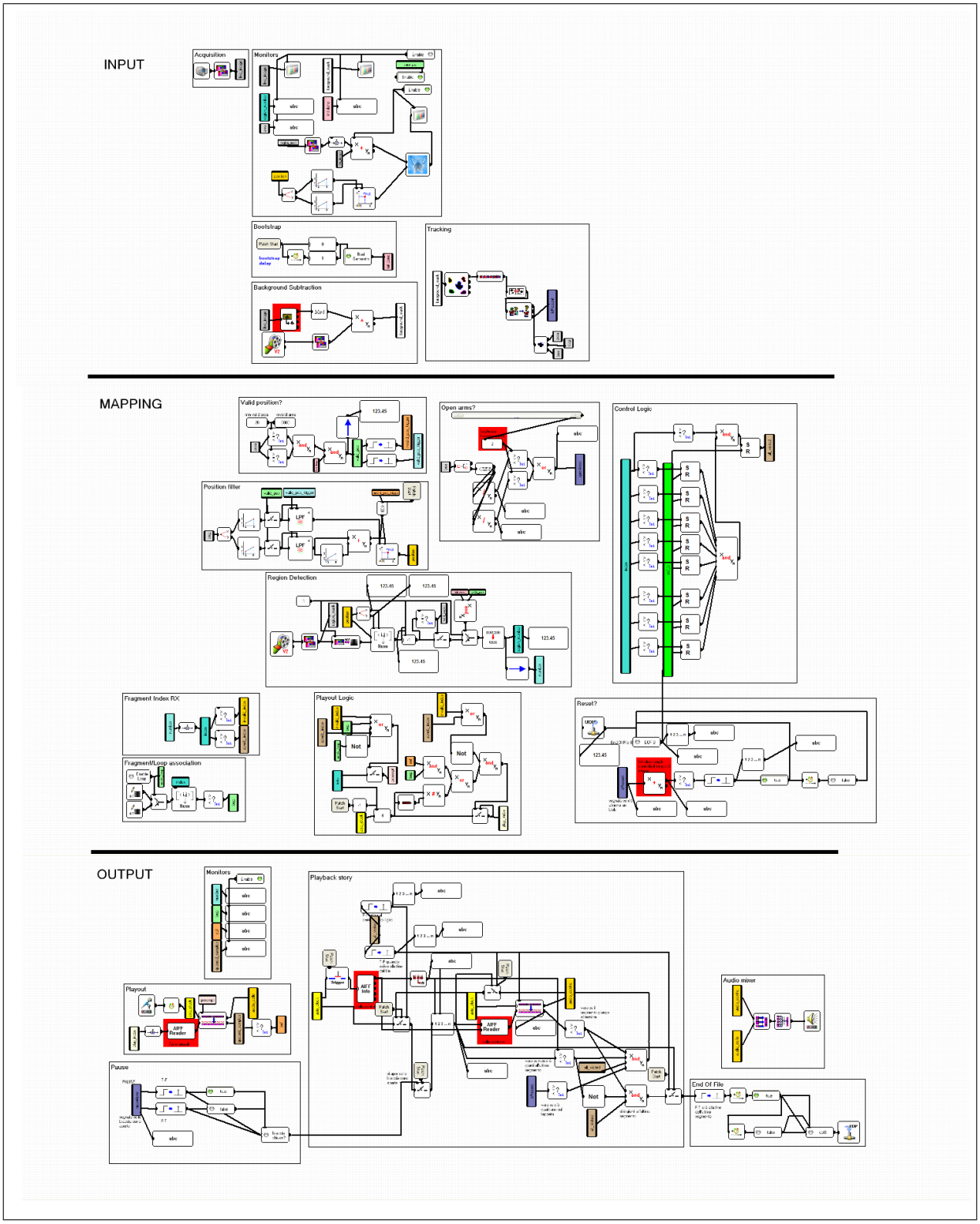


Figure 3. The patch developed in the EyesWeb environment.

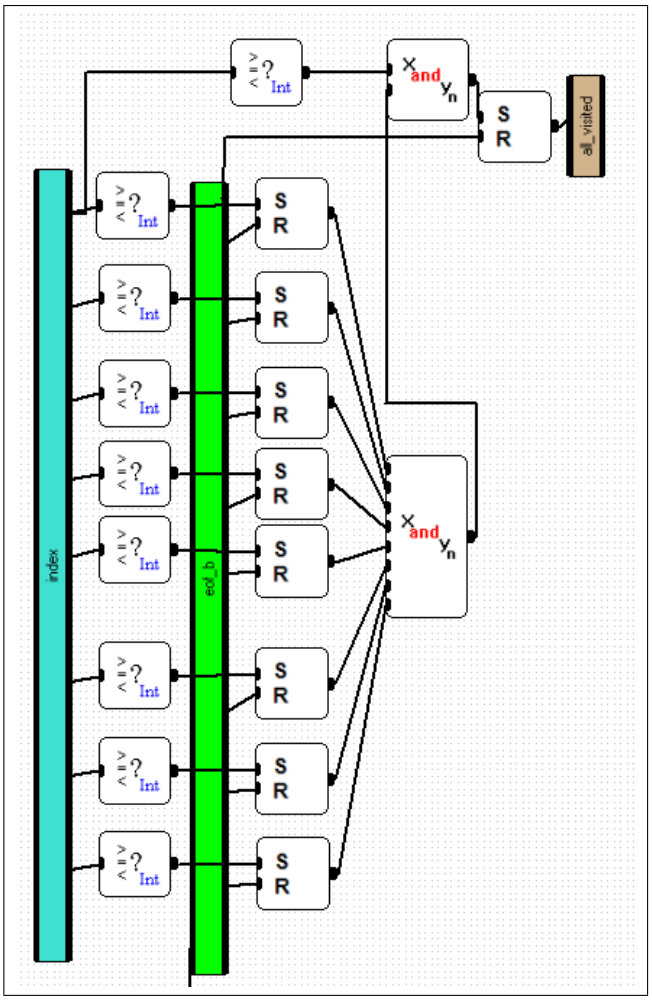


Figure 4. The subpatch that manages the transition among the operative states of the application.

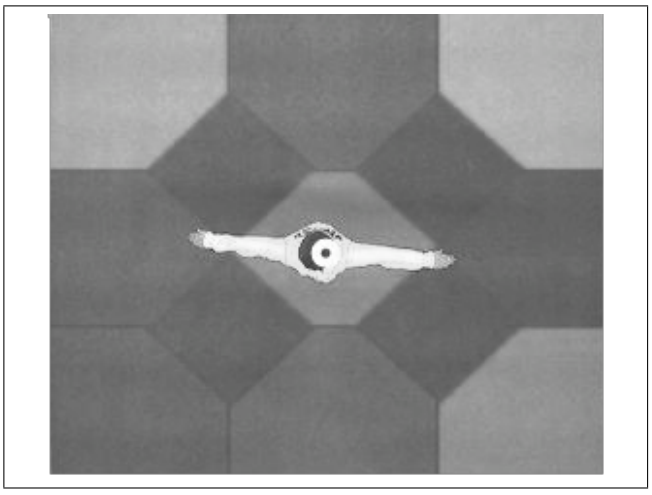


Figure 5. The Contraction Index allows the user to stop the reproduction of the story simply by opening her arms.

Figure 6. Learning English.

- in first grade classes - for a Continuity Project between Nursery School (10 hours) and Primary school and for the integration of a child with autism (3 hours);
- in second grade classes - for the Project "Infolibro" (aimed at the writing of a book for children, 6 hours) and for the development of the communication skills of a child with severe autism (5 hours);
- in third grade classes - for history (2 hours);
- in fourth grade classes - for music (6 hours);
- in fifth grade classes - for two children with dyslexia (7 hours).
- all the classes use the *Stanza Logo-Motoria* for English (40 hours).

We will now describe two particular learning pathways: learning English and the study of science for dyslexic children.

6.1 Learning of a foreign language

From March through May the school organized English language courses held by a mother tongue English teacher; the courses take place every Thursday afternoon and are organized on two levels of complexity. Teaching English through *Stanza Logo-Motoria* has one main goal: to accustom the children to listening to phonemes that are different from the ones of their own language. In this way learning a foreign language increases the awareness of non linguistic tools to communicate, e.g. body language, images, sounds, and symbols for their value across all cultures. The *Stanza Logo-Motoria*, used as a language vehicle, allows activities to be structured so that genuine connections with other disciplines are emphasized. Thus, the enhancement of cognitive skills and the Integrated Learning is encouraged. Segments of speech and phrases become part of the pupils' language knowledge simply because it is used to carry out a task. So speaking English becomes spontaneous, fun, and natural (see fig. 6).

Many languages and methodologies are involved in this system of teaching:

- musical language: enjoying music, acquiring skills to discriminate sound and timbre characteristics of musical instruments;
- theatrical language: characters animation, mime acts and conditions;
- use of new technologies;
- use of community languages to learn disciplines: C-LIL, Content and Language Integrated Learning is a teaching methodology that involves learning a discipline through a foreign language.

Children explore the eight peripheral areas and they memorize the spatial coordinates of sounds (noises, environmental sounds or music); then, entering the central area, they activate a sound reproduction of a story in English. Children must then listen carefully in order to trigger the correct sound at the right time within the story and they have to move in the space searching for the sounds suggested by the story. Sometimes, in one or more peripheral zones, we include several sequences of the story without nouns, adjectives or verbs that the child must insert by saying them aloud.

High concentration in listening and body movement in space ensure effective learning. Even after some time, children are able to recall the exact contents learnt during a particular session within the *Stanza Logo-Motoria*. From analysis of video recording, together with the teachers we observed that this method of teaching increases the motivation to listen and consequently to learn new words and phrases. The evaluation tests carried out successively by the teachers show that children master the contents assimilated through the *Stanza Logo-Motoria*.

6.2 Support for dyslexia

Every year the school is presented with cases of dyslexia. Teachers need to use compensatory and dispensatory measures in order to facilitate learning on the part of dyslexic children and apply a specific evaluation. For this reason *Resonant Memory* is used also by dyslexic children as an alternative tool for learning curriculum subjects. In this case the *Stanza* is used as follows:

- in the first session the text to study is divided into sound sequences, each sequence corresponds to an area of the room; each area of the room is set up with an empty billboard; the students enter the area, one at a time, they listen to the text sequence and put the correct images on the billboard as suggested by the text;
- in the second session the students enter the area, one at a time, they listen to the text sequence observing the images previously positioned onto the poster then they repeat the content with the help of the images;
- the teachers orally assess whether the students' performance has improved after using the *Stanza*.

The school is also attended by a child with visual impairment and one with behavioral disorders who regularly participate in activities organized in *Stanza*. These children integrate with the group of peers and they succeed in the tasks because this tool requires learning skills that they actually possess: there is no text to read, no written questions to be answered, no strings of mathematical operations to solve but there are sounds to listen to and movements to perform.

6.3 Major results

In teaching with the *Stanza Logo-Motoria* it is possible to observe immediately great involvement on the part of the children, high motivation, and extension of the period of attention. All the school children, including the disabled people, are enthusiastic to use the *Stanza*. The use of *Stanza Logo-Motoria* with disabled children offers them the possibility to:

- enhance their communication skills providing alternatives or additions to the mode of communication already in place;
- encourage interaction with others and with the environment;
- extend the time of attention;
- develop expressivity of gesture;
- improve autonomy through production of intentional actions.

In particular using the *Stanza Logo-Motoria*, as a tool for learning for dyslexic students, allows to bypass the written code which, in this case, represents an obstacle to understanding. Children show greater awareness of contents, they regain motivation to learn and they are excited to present what they have learnt to the teacher and to their classmates. Accordingly, children get top scores in the assessment tests of subjects studied in the *Stanza*. Teachers are highly motivated to carry out activities in *Stanza*, indeed they spontaneously propose topics and paths to explore.

Between January and May we have had the chance to assess that:

- Compared to the first term (from September to December when the *Stanza Logo-Motoria* wasn't in use) pupils with dyslexia improved their performance in evaluation tests as well as their approach to all other subjects showing an overall greater self-confidence. Teachers have also noticed that in the *Stanza* these children are able to summarize the contents with greater security linking the various subjects; a performance this that they were not able to do only working in the classroom.
- Regarding English learning, performed in *Stanza*, the teacher pointed out that, thanks to the total and immersive sound perception, the children have a general better understanding of English as well as an improved pronunciation and oral production.

These assessments were carried out by the teacher through:

- direct observation;
- analysis of video recording made inside the *Stanza*;
- school assessment tests.

We certainly believe that in time it is possible to gather more evaluative information; this is a goal that we aim to achieve in the coming years. We intend to create a control group that will allow us to assess over time whether there is a difference between the learning of children who use the *Stanza* and those who have never used it. We will also detect whether a change occurs between the average marks registered in the past (when the *Stanza* was not there) and those of following years (when the *Stanza* was used).

7. CONCLUSIONS

Experiments conducted on 170 pupils of a Primary School have shown that the *Stanza Logo-Motoria* allows users to gradually face many organization tasks across progressively complex experiences. To provide a more complete sensory experience, we plan to modify the system adding a visual feedback and a spatialized rendering of sound.

We are also thinking of providing further opportunities for interaction among children by designing a modality in which the mapping of the gestures of one user depend on the gestures of another. This feature is essential when the user has relationship difficulties such as those typical of autism.

Multimedia is a great opportunity for the evolution of the Educational System as the children are able to use all the elements and tools to build a positive relationship with the world and with themselves. Children who are not yet adults, bound by the self-control which is typical of written communication, are human beings in evolution that receive all the relevant information to put themselves in relation with the world.

8. REFERENCES

[1] C. J. Meijer, V. Soriano, and A. Watkins, *Special Needs Education in Europe*. European Agency for Development in Special Needs Education, in collaboration with Eurydice, 2003.

[2] G. Olimpo, "I nuovi ambienti interattivi per l'apprendimento," *ITD-CNR*, 1996.

[3] H. Gardner, *Frames of Mind: The Theory of Multiple Intelligences*. Basic, 1983.

[4] H. Gardner, *Educazione e sviluppo della mente. Intelligenze multiple e apprendimento.* Centro Studi Erickson, 2005.

[5] G. Rizzolatti and L. Vozza, *Nella mente degli altri. Neuroni specchio e comportamento sociale.* Zanichelli, 2008.

[6] M. Leman, *Embodied Music Cognition and Mediation Technology*. The MIT Press, 2007.

[7] F. Morganti and G. Riva, *Conoscenza, comunicazione e tecnologia. Aspetti cognitivi della Realtà Virtuale.* LED Edizioni Universitarie, 2006.

[8] J. Bruner, *Toward a theory of instruction*. Belknap Press of Harvard University Press, 1966.

[9] M. Krueger, *Responsive Environments. The New Media Reader*. The MIT Press, 1977.

[10] M. Krueger, T. Gionfriddo, and K. Hinrichsen, "Videoplace - an artificial reality. human factors in computing systems," *ACM press*, 1985.

[11] R. Gehlhaar, "Sound=space: An interactive musical environment," *Contemporary Music Review*, vol. 6, no. 1, pp. 59–72, 1991.

[12] R. Gehlhaar, L. M. Girao, and P. Rodriguez, "Cadaremi - an educational interactive music game," *ICDVRAT with Art Abilitation, Maia, Portugal*, 2008.

[13] A. Camurri, G. D. Poli, M. Leman, and G. Volpe, "Toward communicating expressiveness and affect in multimodal interactive systems for performing art and cultural applications," *IEEE Multimedia*, 2005.

[14] F. Laban, *Effort*. Macdonald and Evans Ltd., 1977.

[15] H. Wallbott, "Bodily expression of emotion," *European Journal of Social Psychology*, 1998.

[16] M. D. Meijer, "The contribution of general features of body movement to the attribution of emotions," *Journal of Nonverbal Behavior*, 1989.

[17] J. C. R. T. Boone, "Children's decoding of emotion in expressive body movement: The development of cue attunement," *Developmental Psychology*, 1998.

[18] A. Camurri and T. B. Moeslund, *Visual Gesture Recognition. From motion tracking to expressive gesture*. Appears as chp. 10 in the book Musical Gestures. Sound, Movement, and Meaning. Rolf Inge Godøy and Marc Leman (Eds.). Published by Routledge, ISBN: 9780415998871, 2010.

[19] T. Horprasert, D. Harwood, and L. Davis, *A Robust Background Subtraction and Shadow Detection*, vol. 1. In 4th ACCV, Taipei, Taiwan, 2000.

SOUND TEXTURE SYNTHESIS WITH HIDDEN MARKOV TREE MODELS IN THE WAVELET DOMAIN

Stefan Kersten, Hendrik Purwins
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{stefan.kersten,hendrik.purwins}@upf.edu

ABSTRACT

In this paper we describe a new parametric model for synthesizing environmental sound textures, such as running water, rain, and fire. Sound texture analysis is cast in the framework of wavelet decomposition and multiresolution statistical models, that have previously found application in image texture analysis and synthesis. We stochastically sample from a model that exploits sparsity of wavelet coefficients and their dependencies across scales. By reconstructing a time-domain signal from the sampled wavelet trees, we can synthesize distinct but perceptually similar versions of a sound. In informal listening comparisons our models are shown to capture key features of certain classes of texture sounds, while offering the flexibility of a parametric framework for sound texture synthesis.

1. INTRODUCTION

Many sounds in our surroundings have textural properties—yet *sound texture* is a term difficult to define, because these sounds are often perceived subconsciously and in a context-dependent way. Sound textures exhibit some of the statistical properties that are normally attributed to noise, but they arguably do convey information; not so much in an information theoretic sense, but rather as a carrier of emotional and situational percepts [1]. Indeed, sound texture—often denoted *atmosphere*—forms an important part of the sound scene in real life, in movies, games and virtual environments.

In this work our goal is to synthesize environmental sounds with textural properties, such as running water, waves, fire, crowd noises, etc. Eventually, we intend to provide a building block for an application that automatically generates soundscapes for virtual environments. Our work is in the context of stochastic sound synthesis: given a *textural* analysis or target sound with statistical characteristics sufficiently close to stationarity, we want to synthesize stochastic variations that are perceptually close in their characteristics to the original but are not mere reproductions. In a *data-driven* approach we build a model by statistical signal analysis. The distributions captured by the model are then used to synthesize perceptually similar sounds by stochastic sampling.

Previous research suggests that textural sounds are perceived by human listeners in terms of the statistical properties of constituent features, rather than by individual acoustic events [2, 3]. The ability to model texture in a statistical sense, without detailed knowledge or assumptions about the structure of the source material, leads to several desirable properties that a texture model should possess:

- **Compactness of representation**: The model should require significantly less parameters than the original coded audio.
- **Statistical properties**: The signal statistics should be discoverable using a limited amount of training data.

In general a texture model for synthesis can be split in an analysis part and the actual synthesis part. The goal of the analysis phase is to estimate the joint statistics of signal coefficients in some decomposition space and combine them in a parametric or non-parametric model by statistical analysis. For audio signals, we typically need to estimate not only the *vertical* coefficient relationships, i.e. their interdependencies across the frequency axis, but also their *horizontal* dependencies across time. During the synthesis phase, a new time series of decomposition coefficients is generated by stochastic sampling from the model. If our model sufficiently captured the structural coefficient dependencies, then after transforming the sampled coefficients to the time domain, we obtain a signal that perceptually resembles the original but is not exactly the same.

Multiresolution (MR) signal analysis methods, and in particular the discrete wavelet transform, have been shown to be well suited for modeling the dynamics of sound textures, where important perceptual details are present in various frequency bands and on different time scales [4, 5, 6]. Even though the wavelet transform can be considered almost sparse for many natural signals [8], the coefficients retain inter- and intra-scale dependencies that have to be taken into account in a statistical decomposition and synthesis model. It has been shown that for natural signals like 2D images, the wavelet coefficients themselves are non-Gaussian, but approximately Gaussian conditioned on their context, i.e. neighboring coefficients in scale and location [7]. The hidden Markov tree (HMT) model [8] is a parametric statistical model, that captures inter-scale dependencies and is particularly suited to be applied to tree structured data like wavelet transform coefficients.

While previous approaches to sound texture synthesis have mostly been based on non-parametric density estima-

tion techniques—see Section 2 for an overview—the HMT has been successfully applied to a wide range of image processing problems, leaving room for speculation that it will also be applicable to sound texture modeling. It thus forms the basis of our approach to sound texture synthesis. Our model is similar to previous work in that we also use the wavelet transform for multiresolution signal analysis and perform density estimation in wavelet decomposition trees. Our estimator, however, instead of being based on nonparametric density estimation, explicitly casts the wavelet coefficient statistics and their interdependencies in a graphical model within the maximum likelihood framework. As with parametric models in general, when the modeling assumptions match the signals being modeled fairly well, we can gain from a principled probabilistic approach, e.g. by introducing priors, dealing with missing data and performing inference.

The rest of the paper is structured as follows: In Section 2 we give an overview of current approaches to sound texture modeling and multiresolution statistical analysis. In Section 3 we introduce the basic building blocks of our texture model, the discrete wavelet transform and the hidden Markov tree model and how these fit together in a synthesis model. In Section 4 we present results of natural sound textures synthesized from our model and finally draw some conclusions and mention possible future work in Section 5.

2. RELATED WORK

While image texture modeling has been under active investigation for at least 35 years, sound texture modeling has begun to find a similarly thorough treatment only relatively recently; for an overview with a focus on synthesis see [9].

Many approaches to sound texture modeling have been heavily inspired by methods originally developed for modeling texture images. In [4] the authors describe a nonparametric sound texture model that learns conditional probabilities along paths in a wavelet decomposition tree. Path probability densities are estimated first for inter-scale coefficient dependencies and in a second step for intra-scale predecessor probabilities. In a similar fashion, [5] estimate the sound texture statistics on wavelet tree coefficients by kernel density estimation and histogram sampling, inspired by the approach taken by Efros and Leung for image texture synthesis [10]. The authors report improved results compared to the ones obtained by [4], but didn't conduct a conclusive quantitative evaluation.

A large body of research is devoted to the field of multiresolution statistical models, and in particular MR Markov models, for a comprehensive overview see [11]. The hidden Markov tree model has been applied to a wide range of problems in image and signal processing, such as denoising [8, 12, 13, 14] and texture classification and synthesis [15].

3. METHODS

3.1 The discrete wavelet transform

The discrete wavelet transform decomposes a one- or multidimensional signal $z(t)$ into atoms of shifted and dilated bandpass wavelet functions $\psi(t)$ and shifted versions of a lowpass scaling function $\phi(t)$, i.e. the signal is represented on multiple time scales K and frequency scales J:

$$\begin{aligned} \psi_{J,K}(t) &\equiv 2^{-J/2}\psi(2^{-J}t - K) \\ \phi_{J_0,K}(t) &\equiv 2^{-J_0/2}\phi(2^{-J_0}t - K) \\ J, K &\in \mathbb{Z} \end{aligned} \tag{1}$$

When designed with certain constraints, the wavelet and scaling functions form an orthonormal basis with the following signal representation [16]:

$$\begin{aligned} z(t) &= \sum_K u_K \phi_{J_0,K}(t) + \sum_{J=0}^{J_0} \sum_K w_{J,K}\psi_{J,K}(t) \\ u_K &= \int z(t)\phi^*_{J_0,K}(t)dt \\ w_{J,K} &= \int z(t)\psi^*_{J,K}(t)dt \end{aligned} \tag{2}$$

where $*$ denotes complex conjugation. u_K and $w_{J,K}$ are called *scaling* and *detail coefficients*, respectively. In (1) and (2), J specifies the scale or resolution of analysis – the smaller J, the higher the resolution. J_0 is the lowest level of resolution, where the analysis yields both detail coefficients and scaling coefficients. In the case of audio signals, K denotes the temporal support of analysis, i.e. the amount of time a wavelet $\psi(t)$ is shifted from its support at time zero. The *detail coefficient* $w_{J,K}$ measures the signal content at time $2^J K$ and frequency $2^{-J} f_0$, where f_0 is the wavelet's center frequency. The *approximation coefficient* u_K measures the local mean at time $2_{J_0}K$. Following [8] and in order to reduce notational overhead, we will adopt a simplified indexing scheme for basis functions of the decomposition and the resulting coefficients: instead of indexing by scale J and shift K, we will use a one-dimensional mapping $J, K \mapsto \mathbb{Z}$, where the indices $i \in \mathbb{Z}$ have a fixed but unspecified ordering.

In practice, the DWT can be implemented with a pyramidal filterbank algorithm, where the signal is recursively split into lowpass and highpass filter responses, that together form a quadrature mirror filter pair. Both responses are downsampled by two; the highpass response forms the detail coefficients, while the lowpass response is used for further recursive analysis until a maximum depth is reached.

Due to the recursive structure of the DWT and the shift and dilation relations based on powers of two, the decomposition can be represented as a *forest* (list) of binary trees, where each coefficient in scale J has two children in the next finer resolution scale. At the coarsest level of detail the signal is represented as pairs of detail and approximation coefficients, at which a binary tree of detail coefficients is rooted. The decomposition of the time-frequency

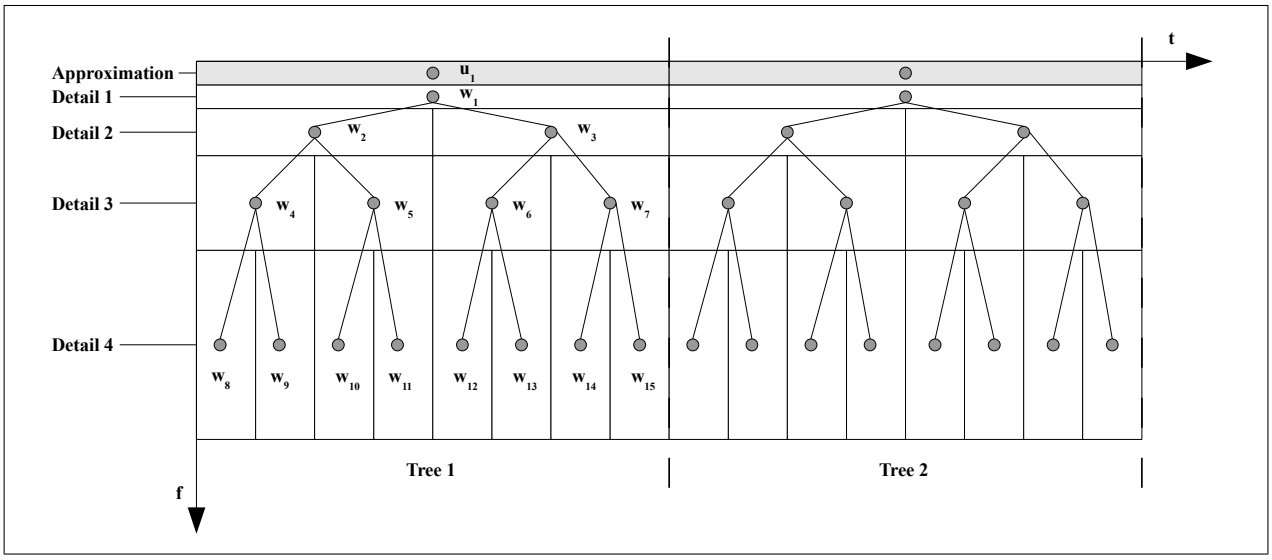


Figure 1: Four level wavelet time-frequency decomposition. Shown are two consecutive trees with frequency running downward and time to the right.

plane into multiple wavelet trees is shown graphically in Fig. 1.

The recursive shifting and dilation performed by the DWT is also the reason for some desirable properties for the analysis of natural sounds [8]: *Locality*, i.e. each bandpass atom ψ_i is localized in both time and frequency, which implies *multi-resolution*, i.e. a nested set of scales in time and frequency is analyzed and *compression*, i.e. the wavelet coefficient matrices of many real-world signals are nearly sparse. These properties are desirable for the goal of estimating the statistics of a wavelet decomposition, as will become evident in the next paragraph.

3.2 Hidden Markov Tree Models

In general, a hidden Markov model introduces hidden *state variables* that are linked in a graphical model with Markov dependencies between the states, as is the case for the widely used hidden Markov model (HMM). Often the hidden states can be viewed as encoding a hidden physical *cause* that is not directly observable in the signal itself or its transformation in feature space.

In our research we focus on MR Markov processes that are defined on pyramidally organized binary trees, in particular the hidden Markov tree model. In this model, each node in a wavelet decomposition tree is identified with a mixture model, i.e. a hidden, discrete valued state variable with M possible values and an equal number of parametric distributions (usually Gaussians) corresponding to the individual values of the hidden state (Fig. 2).

Instead of assuming a Markov dependency on the wavelet coefficients as in parametric estimation methods (see Section 2), the HMT model introduces a first order Markov dependency between a given hidden state and its children. In other words and for the example tree in Fig. 2, given their parent state variable s_1, the subtrees rooted at the children s_2 and s_3 are conditionally independent. Similarly, since the wavelet coefficients are modeled by a distribu-

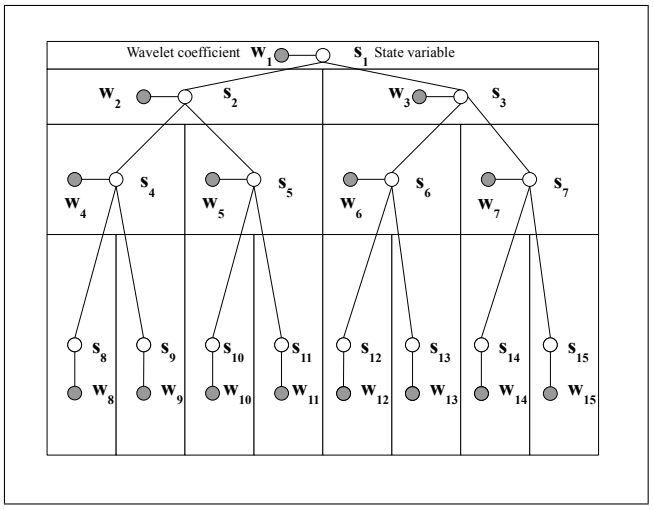


Figure 2: Hidden Markov tree model. Associated with each wavelet coefficient at a certain position in the tree structure (black node) is a hidden state variable (white node), that indexes into a family of parametric distributions.

tion that is only dependent on the node's state, w_2 and w_3 are also independent of the rest of the tree given their parent state s_1. Given enough data for parameter estimation and by increasing the number of states M, we can approximate the marginal distributions of wavelet coefficients to arbitrary precision. This allows us to model marginal distributions that are highly non-Gaussian, but are Gaussian conditioned on the parent state variable.

Even though the wavelet transform can be considered a de-correlator and the decomposition is sparse, the wavelet coefficients of real-world signals can not be considered independent, and there remain inter-coefficient dependencies that need to be taken into account in a statistical model. Fig. 3 shows histograms of wavelet coefficients for a particular scale in natural sound signals. These distributions

are sharply peaked around zero with long symmetric tails, corresponding to the sparsity property of the wavelet transform: there's a large number of small and only a few large coefficients.

For modeling the non-Gaussian wavelet coefficient marginal statistics, we use a two-state Gaussian mixture model, where one state encodes the peaked distribution of small coefficients and the other state encodes the tailed distribution of high-valued coefficients. For wavelet coefficients $\mathbf{w} = (w_1, \ldots, w_N)$ and hidden states $\mathbf{s} = (s_1, \ldots, s_N)$, the HMT is defined by the parameters

$$\begin{aligned} \theta = \{p_{s_1}(m), \epsilon^{mr}_{i,p(i)}, \mu_{i,m}, \sigma^2_{i,m}\} \\ m, r \in \{0,1\}, 1 \leq i \leq N \end{aligned} \tag{3}$$

with:

- $p_{s_1}(m) = P(s_1 = m)$, the probability of the root node s_1 being in state m,
- $\epsilon^{mr}_{i,p(i)} = P(s_i = m | s_{p(i)} = r)$, the conditional probability of child s_i being in state $m \in \{0,1\}$ given the parent $s_{p(i)}$ is in state $r \in \{0,1\}$,
- $\mu_{i,m}$, the mean of the wavelet coefficient w_i given s_i is in in state m $(1 \leq i \leq N)$ and
- $\sigma^2_{i,m}$, the variance of the wavelet coefficient w_i given s_i is in state m $(1 \leq i \leq N)$.

3.2.1 Training of the HMT

In order to find the best parameters fitting a given source sound, we update the model parameters θ given the training data $\mathbf{w} = \{w_i\}$ (a forest of binary wavelet trees, see Section 3.1) using a *maximum likelihood* (ML) approach. The *expectation maximization* (EM) framework provides a solid foundation for estimating the model parameters θ and the probabilities of the hidden states $\mathbf{s}$ and has been formulated for wavelet-based HMM's in [8]. The objective function to be optimized is the *log-likelihood function* $\ln f(\mathbf{w}|\theta)$ of the wavelet coefficients given the parameters. The EM algorithm iteratively updates the parameters until converging to a local maximum of the log likelihood.

In the following we provide a schematic description of the algorithm. More information on how we initialized our models and on the convergence criterion can be found in Section 3.3.

1. **Initialization**
   (a) Select an initial model estimate θ^0,
   (b) Set iteration counter $l = 0$.
2. **E step**: Calculate $P(\mathbf{s}|\mathbf{w}, \theta^l)$, the probability of the hidden state variables $\mathbf{S}$, yielding the expectation

$$Q(\theta|\theta^l) = \sum_{\mathbf{s} \in \{0,1\}^N} P(\mathbf{s}|\mathbf{w}, \theta^l) \ln f(\mathbf{w},\mathbf{s}|\theta). \tag{4}$$

3. **M step**: Update parameters θ, in order to maximize $Q(\theta|\theta^l)$:

$$\theta^{l+1} = \arg\max_{\theta} Q(\theta|\theta^l). \tag{5}$$

4. **Convergence**: Set $l = l+1$. If converged, then stop; else, return to E step.

3.2.2 E Step

The formulas for solving the hidden Markov tree E step presented in [8] are susceptible to underflow due to the multiplication of a large number of probabilities smaller than one. In [13] the authors develop an algorithm that is immune to underflow and computes the probabilities $p(s_i = m|\mathbf{w}, \theta)$ directly, instead of deriving them from $p(s_i = m, w = \mathbf{w})$. The probabilities $p(s_i = m, s_{\rho(i)} = n|\mathbf{w}, \theta)$, needed for computing the conditional state probabilities, can also be extracted directly from their algorithm. Similar to the original algorithm in [8], the above-mentioned probabilities are computed in separate *upward* and *downward* recursions, comparable to the computation of *forward* and *backward* variables in conventional hidden markov models. The algorithm has a slightly higher computational complexity than the one in [8], although it is still linear in the number of observation trees. For a more thorough treatment of the computations involved see [13].

3.2.3 M Step

After having calculated $P(\mathbf{s}|\mathbf{w}, \theta^l)$ in the E step, the M step consists in straight-forward closed-form updates of the conditional state probabilities and the parameters of the observation distributions.

First we calculate the probability of node i being in state m

$$p_{s_i}(m) = \frac{1}{K} \sum_{k=1}^{K} P(s_i^k = m|\mathbf{w}^k, \theta^l) \tag{6}$$

Then we update the model parameters by averaging over the quantities computed in the E-step for each of the K training examples:

$$\epsilon^{mr}_{i,p(i)} = \frac{\sum_{k=1}^{K} P(s_i^k = m, s_{p(i)}^k = r|\mathbf{w}^k, \theta^l)}{K p_{s_{p(i)}}(r)} \tag{7}$$

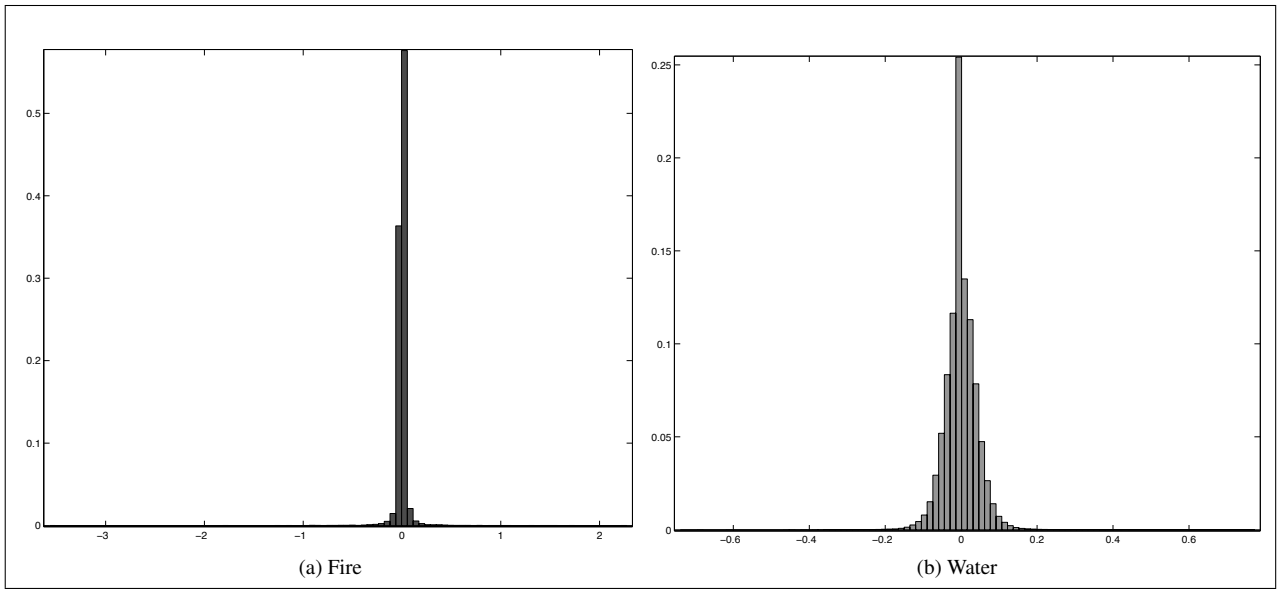
$$\mu_{i,m} = \frac{\sum_{k=1}^{K} w_i^k p(s_i^k = m|\mathbf{w}, \theta^l)}{K p_{s_i}(m)} \tag{8}$$

$$\sigma^2_{i,m} = \frac{\sum_{k=1}^{K} (w_i^k - \mu_{i,m})^2 P(s_i^k = m|\mathbf{w}^k, \theta^l)}{K p_{s_i}(m)} \tag{9}$$

3.3 Application to Sound Texture Synthesis

In this section we describe how the hidden Markov Tree model is adapted to a sound texture synthesis application. [1]

[1] All of the algorithms used in this work were implemented in the functional programming language Haskell and a link for downloading the package can be found at `http://mtg.upf.edu/people/skersten?p=Sound%20Texture%20Modeling`



(a) Fire (b) Water

Figure 3: Histograms of wavelet coefficients on the first (finest) scale of a five-level decomposition for two natural sounds, fire (left) and running water (right). The wavelet coefficient statistics for fire are clearly non-Gaussian, while for running water the statistics approach the Gaussian distribution.

3.3.1 Wavelet decomposition

The signal is first decomposed into wavelet coefficients on different scales, using Daubechies wavelet functions with five and ten vanishing moments [16].

The wavelet decomposition yields a forest of binary trees, each rooted at one of the corse scale wavelet coefficients. The length of the window corresponding to an individual tree depends on the depth of the wavelet decomposition. In our experiments, we chose a decomposition depth of 15 and 16, respectively, which corresponds to a context length of $2^{15} = 32768$ (or 0.74s at a sample rate of 44100) and $2^{16} = 65536$ (or 1.5s). In order to yield an even number of trees, the signal was truncated to an integer multiple of the context length in samples, i.e. from a signal of length n with a decomposition depth of m we used the first $\lfloor \frac{n}{2^m} \rfloor * 2^m$ samples. We worked exclusively with monophonic sounds and extracted the left channel from sounds that have originally been recorded in stereo.

3.3.2 Model construction

The wavelet decomposition tree structure is mapped to a HMT by associating each coefficient i with a hidden state variable s_i that can take one of two discrete values, depending on the value of the parent's state variable $s_{\rho(i)}$ (see Section 3.2). Together with one normal observation distribution per state value, each node forms a Gaussian mixture model that approximates the statistics of a coefficient at a certain position in the tree. The model has the same number of nodes as a single wavelet tree and all the trees in a decomposition forest are regarded to be independent samples from the same underlying distribution which corresponds to the parameter tying *across trees* described in [8].

In order to simplify the model, we don't take the approximation coefficient corresponding to each corse scale coefficient into consideration, although an extension to a two-dimensional Gaussian mixture for the root node would be straight-forward.

3.3.3 Model initialization and training

Since the EM algorithm only converges to a local minimum of the likelihood function (see Section 3.2), it is important to find a good initial estimate of the model's parameters. Following [17], we initialize the conditional state probabilities and the Gaussian distribution parameters by fitting a two-state Gaussian mixture model (GMM) to each level of the wavelet decomposition (the corresponding levels of all trees are concatenated). Once the GMM parameters have been found by the EM algorithm for Gaussian mixtures [18], an initial estimate of the conditional state probabilities $\epsilon^{mr}_{i,p(i)}$ is found by averaging over the number of joint state occurrences for each tree node

$$\epsilon^{mr}_{i,p(i)} = \frac{\#(s_i = m \text{ and } s_{\rho(i)} = n)}{\#(s_{\rho(i)} = n)} \tag{10}$$

During training, each tree of the decomposition forest is presented to the HMT model as an independent training example. In the E-step, the probabilities $P(S_i = m|\theta, w_i)$ and $P(S_i = m, S_{\rho(i)} = n|\theta, w_i)$ are determined as described in Section 3.2.2. The M-step then proceeds to update the model parameters according to (7), averaging over all of the trees in the training set.

We trained our models until the training data log likelihood under the updated model in step $l + 1$ was within a margin t of 0.001 of the log likelihood under the model in step l or when a maximum number n of iterations had been reached:

$$r_l \equiv \frac{\ln f(\mathbf{w}|\theta^{l+1}) - \ln f(\mathbf{w}|\theta^{l+1})}{\ln f(\mathbf{w}|\theta^{l+1})} \tag{11}$$

terminate when $0 \leq t \leq r_l \vee l \geq n$.

3.3.4 Synthesis

Sampling from the model begins by choosing an initial state for the root of the tree based on the estimated probability mass function (pmf) and sampling a wavelet coefficient from the gaussian probability distribution function (pdf) associated with the node and the sampled state. The algorithm proceeds by recursively sampling state pmfs and observation pdfs at each node given the state of its immediate parent. After having sampled a number of trees from the model independently from each other—without any explicit tree sequence model—the resulting forest of binary wavelet coefficient trees is transformed to the time domain by the inverse wavelet transform.

4. RESULTS

For a first qualitative evaluation we selected two textural sounds, *fire* and *running water*, from a commercial collection of environmental sound effects [2].

Fig. 4 shows the spectrograms of the fire and the water sound, respectively (left column). The fire texture is composed of little micro-onsets stemming from explosions of gas enclosed in the firewood. Inter-onset intervals are in the range of a few milliseconds. The background is filled with hisses, little pops and some low frequency noise. The sound of a water stream on the other hand is characterized by its overall frequency envelope with a broad peak below 5 kHz and a narrow peak around 12 kHz, while the fine structure is not clearly discernible in the spectrogram.

Informally evaluating the synthesis results by listening [3] shows that the HMT model is able to capture key dependencies between wavelet coefficients of the textural sounds. In the case of fire, the model built from an analysis with the longer wavelet function with ten vanishing moments is not able to reproduce the extremely sharp transients present in the signal. All three fire reproductions capture the overall perceptual quality of the original. This coherence is ensured by the HMT model by capturing the *across scale* coefficient dependencies. The temporal fine structure however can deviate significantly from the original: In all three cases the onset patterns are denser than in the source sound and lack sequential coherence. This can be explained with the fact that our model doesn't capture temporal, i.e. *within-scale* dependencies of wavelet coefficients explicitly. This missing feature roughly corresponds to the autocorrelation feature found to be important for the perception of textures in both image and sound [15, 3].

Similar to the sounds of fire, the synthesis of the water sound shows an overall similar spectral shape to the original, although an important spectral peak is missing from around 12 kHz and the high frequency content is more noisy in general (Fig. 4). In this sound, clearly noticeable *bubbles* form an important part of the temporal fine structure, and this feature is missing from the synthesis. We attribute this, as in the case of fire, to the missing autocorrelation feature in our synthesis model.

All of the synthesis examples show a repeating pattern with a length close to the wavelet tree size, i.e. directly related to the decomposition depth, although there is some minor within-loop variation. This result is an indication that the model is overfitted to the source material and can be explained with the relatively low number of training examples per tree model (around 7 wavelet trees per 10 seconds of source material). We could alleviate the overfitting effect in two ways: firstly, by using a significantly larger training set, and secondly, by *tying* parameters of correlated wavelet coefficients and thereby reducing the number of states and the number of mixture components. Simply tying parameters within one level of the wavelet decomposition however was found to be inadequate, because temporal fine structure gets lost and the synthesis result resembles a noisy excitation with roughly the spectral envelope of the original.

In order to quantitatively assess the synthesis quality, we conducted a small listening experiment with eleven subjects. We selected three sound examples for each of the five texture classes *applause*, *crowd chatter*, *fire*, *rain* and *running water* from the Freesound database [4]. We trimmed the sounds to the first 20 seconds, selected the left channel and downsampled this sound portion to a uniform sample rate of 22.5kHz. We then trained a model for each of the sounds using a wavelet tree decomposition of a depth of 16, i.e. an analysis frame length of 1.5s, and stopping training after 40 iterations. By sampling from the models we synthesized an eight second audio clip for each original sound file and presented the examples in random order. In a forced choice test, the subjects had to assign each synthesized sound to one of the five texture classes.

Table 1 shows the confusion matrix of the listening experiment and Table 2 lists the per-class accuracy. Apparently our model adequately captures the key perceptual properties of the respective sound classes except in the case of *water* and *rain*. The rain/water confusion can be explained with the missing "larger-scale" fine structure in the water examples (bubbling, whirling) that draws them closer to the noisy nature of the synthesized rain. While *applause* gets confused with rain on a surface because of the perceptual similarity between the micro-onsets that comprise those texture sounds, the vocal quality of the *crowd* chatter is a clearly distinguishing feature, even if poorly synthesized.

5. CONCLUSIONS

In this work we approached the problem of sound texture synthesis by application of a multi-resolution statistical model. Our contribution is a model that is able to capture key dependencies between wavelet coefficients for certain classes of textural sounds. While the synthesis results highlight some deficiencies that need to be addressed in future work, a parametric probabilistic approach to sound texture modeling has important advantages:

[2] *Blue Box SFX*, `http://www.eastwestsamples.com/details.php?cd_index=36`, accessed 2010-04-27.

[3] The synthesis results of our experiments are available on the web for reference, `http://mtg.upf.edu/people/skersten?p=Sound%20Texture%20Modeling`, accessed 2010-06-14

[4] `http://freesound.org`

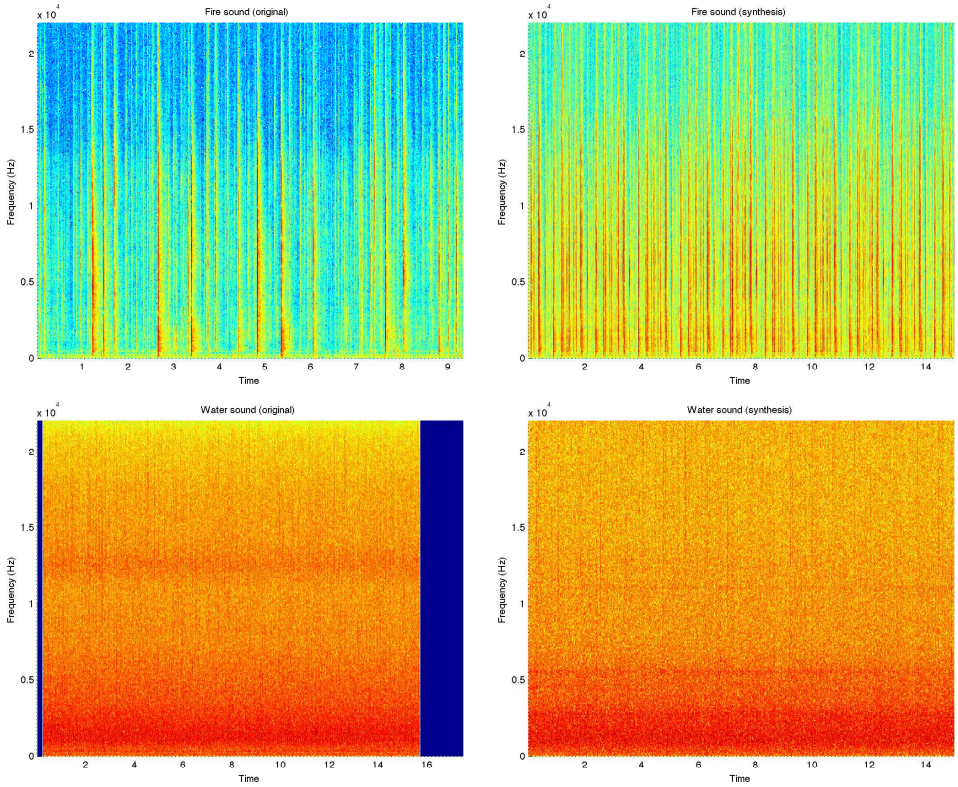


Figure 4: Spectrograms of a fire sound (top left), its synthesis (top right) a water stream sound (bottom left) and its synthesis (bottom right). Both sounds were recorded at a sample rate of 44100 kHz. The spectrum analysis was performed with a window size of 1024 and a hop size of 256.

		Predicted				
		applause	crowd	fire	rain	water
Actual	applause	13	1	0	7	1
	crowd	0	24	1	5	2
	fire	0	0	30	0	3
	rain	1	1	2	17	12
	water	6	0	2	19	6

Table 1: Confusion matrix for the listening experiment's results with five sound classes of three examples each and eleven subjects. Due to an error during the model building process, the applause class contains only two examples. One user classification for the crowd class was not submitted.

	Class				
	applause	crowd	fire	rain	water
Accuracy	0.59	0.75	0.91	0.52	0.18

Table 2: Class accuracies obtained in the listening experiment.

- *Probabilistic priors* can be used to deal with insufficient training data or to expose expressive synthesis control parameters.
- The model can be applied to *inference tasks* like classification, segmentation and clustering.

When comparing the synthesized sounds to their original source sounds it becomes evident that the model fails to capture some features that are crucial for auditory perception of texture, most notably the intra-scale autocorrelation feature. Another major limitation is the inadequate representation of infinite time series, because our model divides the signal into *blocks* of a size determined by the model tree depth, thereby introducing artifacts caused by the position of the signal relative to the beginning and the end of the block.

The most intuitive approach to overcome these limitations is to modify the graphical tree model itself, by allowing additional conditional dependencies between nodes on the same hierarchy level. Because graphs that satisfy certain conditions on their structure, and in particular on the cycles formed by their edges, still allow for efficient parameter estimation in the EM framework—see [19] for a thorough treatment—it is possible to model within-scale coefficient dependencies without resorting to Markov-chain Monte-Carlo or other simulation methods. The significant increase in the number of parameters needs to be addressed by aggressive tying, i.e. by using the same parameters for

a set of variables in the model that exhibit the same statistics. While tying within tree levels yields unsatisfactory results for the model described in this paper, a modified model might be able to capture just enough temporal correlations to make this tying scheme feasible. By explicitly modeling dependencies across time, the wavelet decomposition depth wouldn't be the only way to capture temporal context any longer and could be decreased significantly, resulting in a vastly reduced set of parameters.

6. ACKNOWLEDGMENTS

Many thanks to Jordi Janer, Ferdinand Fuhrmann and the anonymous reviewers for their valuable suggestions and to those who kindly participated in our listening test. This work was partially supported by the ITEA2 Metaverse1 Project [5]. The first author is supported by the FI-DGR 2010 scholarship of the Generalitat de Catalunya, the second author holds a Juan de la Cierva scholarship of the Spanish Ministry of Science and Innovation.

7. REFERENCES

[1] R. M. Schafer, *The Soundscape: Our Sonic Environment and the Tuning of the World*. Destiny Books, 1994.

[2] N. Saint-Arnaud, *Classification of Sound Textures*. Master thesis, Massachusetts Institute of Technology, 1995.

[3] J. H. McDermott, A. J. Oxenham, and E. P. Simoncelli, "Sound texture synthesis via filter statistics," in *Proceedings IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, (New Paltz, NY), Oct. 2009.

[4] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Synthesizing sound textures through wavelet tree learning," *Computer Graphics and Applications, IEEE*, vol. 22, pp. 38–48, July 2002.

[5] D. ORegan and A. Kokaram, "Multi-Resolution sound texture synthesis using the Dual-Tree complex wavelet transform," in *Proc. 2007 European Signal Processing Conference (EUSIPCO)*, 2007.

[6] A. Kokaram and D. O'Regan, "Wavelet based high resolution sound texture synthesis," in *Proc. 31st International Conference: New Directions in High Resolution Audio*, June 2007.

[7] D. D. Po and M. N. Do, "Directional multiscale modeling of images using the contourlet transform," *IEEE Transactions on Image Processing*, vol. 15, no. 6, p. 16101620, 2006.

[8] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal processing using hidden markov models," *Signal Processing, IEEE Transactions on*, vol. 46, no. 4, pp. 886–902, 1998.

[5] http://www.metaverse1.org

[9] G. Strobl, G. Eckel, D. Rocchesso, and S. le Grazie, "Sound texture modeling: A survey," 2006.

[10] A. A. Efros and T. K. Leung, "Texture synthesis by Non-Parametric sampling," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, p. 1033, IEEE Computer Society, 1999.

[11] A. Willsky, "Multiresolution markov models for signal and image processing," *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1396–1458, 2002.

[12] H. Choi, J. Romberg, R. Baraniuk, and N. Kingsbury, "Hidden markov tree modeling of complex wavelet transforms," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 1, pp. 133–136 vol.1, 2000.

[13] J. Durand, P. Goncalves, and Y. Guedon, "Computational methods for hidden markov tree models-an application to wavelet trees," *Signal Processing, IEEE Transactions on*, vol. 52, no. 9, pp. 2551–2560, 2004.

[14] D. H. Milone, L. E. D. Persia, and M. E. Torres, "Denoising and recognition using hidden markov models with observation distributions modeled by hidden markov trees," *Pattern Recogn.*, vol. 43, no. 4, pp. 1577–1589, 2010.

[15] G. Fan and X. G. Xia, "Wavelet-based texture analysis and synthesis using hidden markov models," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 1, p. 106120, 2003.

[16] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988.

[17] G. Fan and X. G. Xia, "Improved hidden markov models in the wavelet-domain," *IEEE TRANS SIGNAL PROCESS*, vol. 49, no. 1, p. 115120, 2001.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning*. Information Science and Statistics, New York, NY, USA: Springer, 2006.

[19] H. Lucke, "Which stochastic models allow Baum-Welch training?," *Signal Processing, IEEE Transactions on*, vol. 44, no. 11, pp. 2746–2756, 1996.

HEAD IN SPACE: A HEAD-TRACKING BASED BINAURAL SPATIALIZATION SYSTEM

Luca A. Ludovico, Davide A. Mauro, and Dario Pizzamiglio
LIM - Laboratorio di Informatica Musicale
Dipartimento di Informatica e comunicazione (DICo)
Universitá degli Studi di Milano, Via Comelico 39/41, I-20135 Milan, Italy
{ludovico,mauro}@dico.unimi.it
http://www.lim.dico.unimi.it

ABSTRACT

This paper discusses a system capable of detecting the position of the listener through a head-tracking system and rendering a 3D audio environment by binaural spatialization. Head tracking is performed through face recognition algorithms which use a standard webcam, and the result is presented over headphones, like in other typical binaural applications. With this system users can choose an audio file to play, provide a virtual position for the source in an euclidean space, and then listen to the sound as if it is coming from that position. If they move their head, the signal provided by the system changes accordingly in real-time, thus providing a realistic effect.

1. INTRODUCTION

3D sound is becoming a prominent part of entertainment applications. The degree of involvement reached by movies and video-games is also due to realistic sound effects, which can be considered a virtual simulation of a real sound environment.

In one of the definitions of Virtual Reality, simulation does not involve only a virtual environment but also an immersive experience (see [1]); according to another author, instead of perception based on reality, Virtual Reality is an alternate reality based on perception (see [2]). An immersive experience takes advantage from environments that realistically reproduce the worlds to be simulated.

In our work, we are mainly interested in audio aspects. Even limiting our goals to a realistic reproduction of a single audio source for a single listener, the problem of recreating an immersive experience is not trivial. With a standard headphones system, sound seems to have its origin inside the listener's head. This problem is solved by binaural spatialization, described in Section 3, which gives a realistic 3D perception of a sound source S located somewhere around the listener L. Nowadays, most projects using binaural spatialization aim at animating S keeping the position of L fixed. Thanks to well known techniques, such a result is quite easy to achieve. However, for an immersive experience this is not sufficient: it is necessary to know the position and the orientation of the listener within the virtual space in order to provide a consistent signal [3], so that sound sources can remain fixed in virtual space independently of head movement, as they are in natural hearing [4].

As a consequence, we will introduce a head-tracking system to detect the position of L within the space and modify the signal delivered through headphones accordingly. The system can now verify the position of S with respect to L and respond to his/her movements.

At the moment, audio systems typically employ magnetic head trackers thanks both to their capability of handling a complete 360° rotation and to their good performances. Unluckily, due to the necessity of complex dedicated hardware, those systems are suitable only to experimental or research applications. But the increasing power of home computers is supporting a new generation of optical head trackers, based primarily on webcams.

This work proposes a low cost spatialization system which only relies on resources available to most personal computers. Our solution, developed with MAX/MSP, is based on a webcam head-tracking system and binaural spatialization implemented via convolution.

The paper is structured as follows. First we will provide a short review of related literature and similar systems. Then the basic concepts about binaural spatialization techniques will be introduced. Finally we will describe the integration of a head-tracking system via MAX/MSP externals - namely the multi-platform, real-time programming environment for graphical, audio, and video processing used to implement our approach - and the real-time algorithms involved in the processing of audio and video streams.

2. RELATED WORKS

We want to present here other similar approaches and projects which served as a basis in the development process. Some concepts, such as "binaural spatialization" will be introduced in the following.

- Binaural Tools: A MAX/MSP patch from the author of CIPIC database that performs binaural panning using Head Related Transfer Function (HRTF)

measurements. The panner takes an input sound file and convolves it with a measured sound response recorded from a selectable angle and elevation. Output can optionally be recorded to a sound file. The program was created based on some parts of Vincent Choqueuse's binaural spatializer for Max/MSP [5]. We started from these works to develop our approach. They are inspiring as they do not use external libraries and rely solely on MAX capabilities. This approach has also some drawbacks. For example, in order to perform spatialization efficiently, other techniques could be used but they should be expressly implemented.

- Spat∼: A Spatial Processor for Musicians and Sound Engineers [6]. Spat∼ is a real-time spatial processing software which runs on the Ircam Music Workstation in the MAX graphical signal processing environment. It provides a library of elementary modules (pan-pots, equalizers, reverberators, etc.) linkable into a compact processor integrating the localization of sound events together with the manipulation of room acoustical quality. This processor can be configured for various reproduction formats over loudspeakers or headphones, and controlled through a higher-level user interface including perceptual attributes derived from psychoacoustical research. Applications include studio recording and computer music, virtual reality or variable acoustics in rooms. The stability and quality of this library could be useful to redesign some structures of our spatializer and achieve better quality and performances.

- bin_ambi: A Real-Time Rendering Engine for Virtual (Binaural) Sound Reproduction [7]. This library is intended for the use with Miller Puckette's open source computer music software Pure Data (PD). The library is freely downloadable and can be used under the terms of GNU General Public License. It provides a simple API easy to use for scientific as well as for artistic projects. In this implementation there is a room simulation with 2 sound objects and a listener. One direct signal and 24 early reflections are calculated and rendered per sound object. The sound rendering based on mirror sources provides models for the early reflections. Each reflection will be encoded into the Ambisonics domain (4th order 3-D) and added to the Ambisonics bus. The listener rotates the whole Ambisonics field, the Ambisonics decoder renders the field into 32 discrete signals of 32 virtual loudspeakers. All 32 speaker signals will be filtered by its HRFT in relation to the left and to the right ear (binaural decoding). Interpolation is one of the critical points of such applications. We can choose an approach like the one proposed here that could give a better interpolation and sound quality but increases the computational complexity of the system.

- 3D-Panner [8]: A SuperCollider-based spatialization tool for creative musical applications. The program spatializes monaural sounds through HRTF convolution, allowing the user to create 3D paths in which the sound source will travel. In 3D Panner the user can easily create unique paths that can range from very simple to very complex. These paths can be saved independently of the sound file itself and applied to any other monaural source. During playback, the sound source is convolved with the interpolated HRTFs in real-time to follow the user-defined spatial trajectory. This project is inspiring for our work because we plan to introduce new features, such as moving sound sources, and we need a way to describe and handle trajectories.

3. BINAURAL SPATIALIZATION

Binaural spatialization is a technique that aims at reproducing a real sound environment using only two channels (like a stereo recording). It is based on the assumption that our auditory system has only two receivers, namely the ears. If it is possible to deliver a signal equal (or nearly equal) to the one which a subject would receive in a real environment, this will lead to the same perception. Our auditory system performs various tasks to obtain a representation of the acoustic environment; most of them are based on the physical parameters of the signal of interest and are called "cues" [9][10].

Binaural spatialization can be achieved through various processes, such as: equalizations and delays, or convolution with the impulse response of the head (HRIR). The latter approach is the one we have followed in our work. In order to obtain these impulses, many experiments involving the use of a dummy head [1] have been made (see i.e. [11]), thus creating databases of impulse responses. Most of them use a fixed distance (usually 1 meter) from S to L, which constitutes a potential limitation.

4. INTEGRATING A HEAD-TRACKING SYSTEM INTO MAX

In our work, we choose to adopt faceAPI, namely an optical face tracking system developed by Seeing Machines [12] that provides a suite of functions for image processing and face recognition encapsulated in a tracking engine. It is a commercial product - freely usable only for research purposes - that implements a head tracker with six degrees of freedom. It can be seen as a "black box" which grants access to tracking data through a simple interface oriented to programming tasks. Basically the engine receives frames from a webcam, processes them and then returns information about the position of the head with respect to the camera.

MAX/MSP is an integrated platform designed for multimedia, and specifically for musical applications [13]. This graphical real-time environment can be successfully used by programmers, live performers, "traditional" musicians, and composers. Within the program objects are also represented like "black boxes" which accept input through their

[1] A dummy head is a mannequin that reproduces the human head.

inlets and return output data through their *outlets*. Programs are built by disposing these entities on a canvas (the *patch*) and creating a data flow by linking them together through *patchcords*.

MAX provides developers with a collection of APIs to create external objects and extend its own standard library [14]. The integration of the head tracker requires to create a base project for MAX (we used the so called "minimum project") and then add references to faceAPI to start developing the external.

When MAX loads an external, it calls its *main()* function which provides initialization features. Once loaded, the object needs to be instanced by placing it inside a patch. Then the external allocates memory, defines inlets and outlets and configures the webcam. Finally, faceAPI engine starts sending data about the position of the head. In our implementation the external reacts only to *bang* messages: [2] as soon as a message is generated, a function of faceAPI is invoked to return the position of the head through *float* variables.

Each MAX object has to be defined in terms of a C structure, i.e. a structured type which aggregates a fixed set of labelled objects, possibly of different types, into a single object. Our implementation presents only pointers to the object outlets in order to directly pass variables to the tracking engine.

```
typedef struct _head {
   t_object c_box;
   void *tx_outlet, *ty_outlet, *tz_outlet;
   void *rx_outlet, *ry_outlet, *rz_outlet;
   void *c_outlet;
} t_head;
```

Such values represent the translation along 3 axes (tx, ty, tz), the orientation of the head in radians (rx, ry, rz) and a confidence value. After their detection, values are sent to their corresponding *outlets* and they are available to the MAX environment. In brief, the headtracker external presents only one inlet that receives bang messages and seven outlets that represent the values computed by the tracking engine.

5. THE "HEAD IN SPACE" APPLICATION

This section aims at introducing the Head in Space (HiS) application for MAX. As discussed in Section 4, we assume that our head-tracking external acts as a black box that returns a set of parameters regarding the position of the head.

In Figure 1 a workflow diagram of the system is shown.

In input, two sets of parameters are available to the system, in order to define: 1. the position of the listener, and 2. the position of the audio source. Given this information, and taking into account also the position of the camera, it is possible to calculate the relative position of the listener with respect to the source in terms of azimuth, elevation and distance. This is what the system needs to choose which impulse response to use for spatialization. Once the correct HRIR is obtained from the database, it is possible to perform convolution between a mono audio signal in input and the stereo impulse response. Since the position both of the listener and of the source can change over time, an interpolation mechanism to switch between two different HRIRs has been implemented.

[2] A bang is a MAX special message that causes other objects to trigger their output.

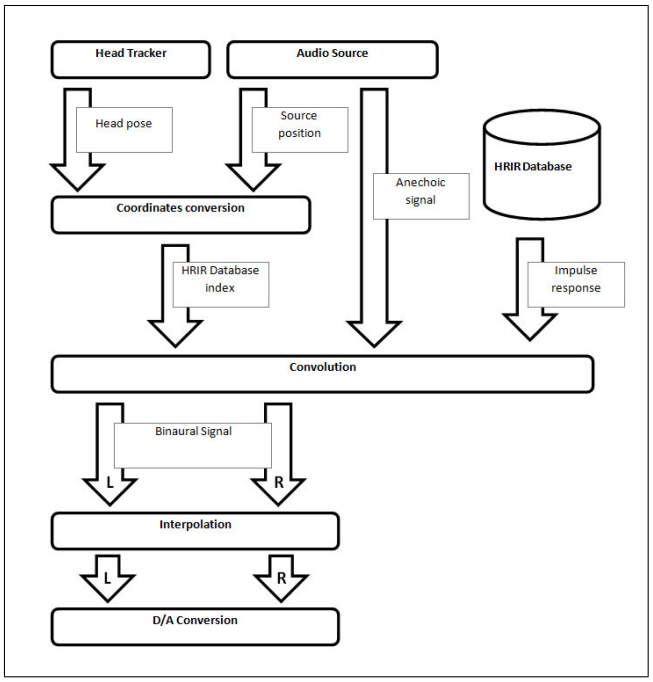


Figure 1. The workflow diagram of the system.

5.1 Coordinates Extraction

The spatializer uses a spherical-coordinates system that has its origin in the center of the listener's head. Source is identified by a distance measure and two angles, namely azimuth on horizontal plane and elevation on median plane. Angular distances are expressed in degrees and stored in the patch through integer variables, whereas the distance is expressed in meters and its value is stored as a float number.

Please note that the head tracker presents coordinates in a cartesian form that has its origin in projection cone of the camera. Thus the representation of coordinates of the spatializer and the one of the head tracker are different and a conversion procedure is needed. The conversion process first performs a rototranslation of the system in order to provide the new coordinates of translation both of the source and of the head inside a rectangular reference system.

Referring to Figure 3, given the coordinates for a generic point P, representing the source in a system $(O_1; X_1, Y_1, Z_1)$, we can determine a set of coordinates in a new cartesian plane $(O_2; X_2, Y_2, Z_2)$ that refers to the position of the head through the relation:

$$V_2 = V_0 + (1 + k) \cdot R \cdot V_1 \qquad (1)$$

where:

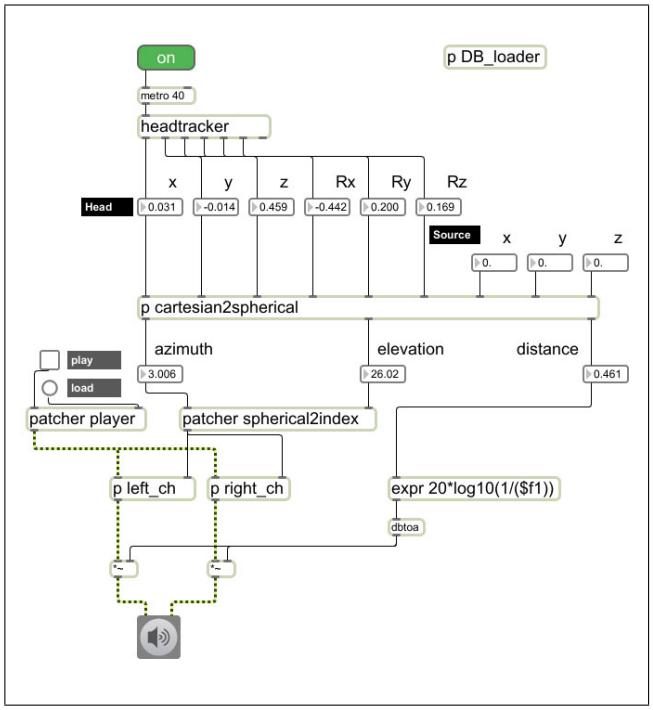


Figure 2. An overview of the patch.

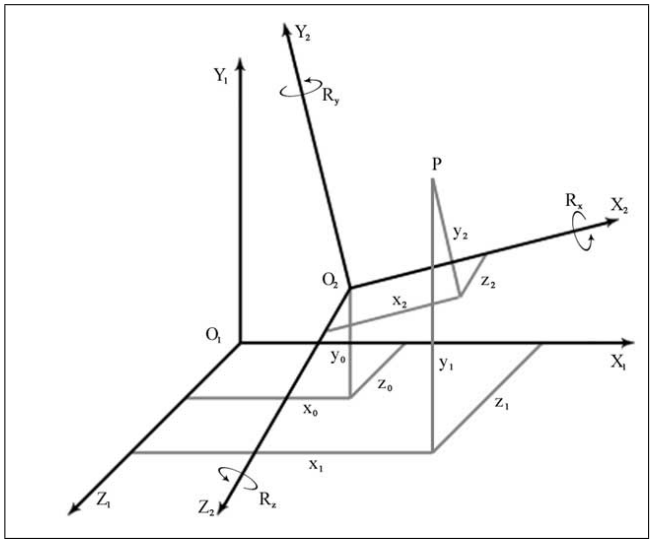


Figure 3. The translation system.

$$V_0 = \begin{Vmatrix} x_0 \\ y_0 \\ z_0 \end{Vmatrix} \quad \text{translation components}$$

$$V_1 = \begin{Vmatrix} x_1 \\ y_1 \\ z_1 \end{Vmatrix} \quad \text{known coordinates of } P \text{ in } O_1$$

$$V_2 = \begin{Vmatrix} x_2 \\ y_2 \\ z_2 \end{Vmatrix} \quad \text{unknown coordinates of } P \text{ in } O_2$$

$$k = 0 \quad \text{scale factor}$$

$$R = R_x \cdot R_y \cdot R_z \quad \text{rotation matrix} \tag{2}$$

R is the matrix obtained by rotating each cartesian triplet with subscript 1 along its axes X_1, Y_1, Z_1 with rotation of R_x, R_y, R_z to displace it parallel to X_2, Y_2, Z_2. Rotation matrixes are:

$$R_x = \begin{Vmatrix} 1 & 0 & 0 \\ 0 & \cos(R_x) & \sin(R_x) \\ 0 & -\sin(R_x) & \cos(R_x) \end{Vmatrix} \tag{3a}$$

$$R_y = \begin{Vmatrix} \cos(R_y) & 0 & -\sin(R_y) \\ 0 & 1 & 0 \\ \sin(R_y) & 0 & \cos(R_y) \end{Vmatrix} \tag{3b}$$

$$R_z = \begin{Vmatrix} \cos(R_z) & \sin(R_z) & 0 \\ -\sin(R_z) & \cos(R_z) & 0 \\ 0 & 0 & 1 \end{Vmatrix} \tag{3c}$$

the product $R_x \cdot R_y \cdot R_z$ is calculated with (4).

We can now derive formulas to calculate the position in the new system:

$$\begin{aligned} x_2 =& (x_0 + x_1)[\cos(R_y)\cos(R_z)] \\ &+ (y_0 + y_1)[\cos(R_x)\sin(R_z) \\ &+ \sin(R_x)\sin(R_z)\cos(R_z)] \\ &+ (z_0 + z_1)[\sin(R_x)\sin(R_z) \\ &- \cos(R_x)\sin(R_z)\sin(R_z)] \end{aligned} \tag{5}$$

$$\begin{aligned} y_2 =& (x_0 + x_1)[\cos(R_y)\sin(R_z)] \\ &+ \\ &(y_0 + y_1)[\cos(R_x)\cos(R_z) \\ &- \sin(R_x)\sin(R_z)\sin(R_z)] \\ &+ (z_0 + z_1)[\sin(R_x)\cos(R_z) \\ &+ \cos(R_x)\sin(R_z)\sin(R_z)] \end{aligned} \tag{6}$$

$$\begin{aligned} z_2 =& (x_0 + x_1)\sin(R_y) \\ &+ (y_0 + y_1)[\sin(R_x)\cos(R_y)] \\ &+ (z_0 + z_1)[\cos(R_x)\cos(R_y)] \end{aligned} \tag{7}$$

Now we can calculate spherical coordinates using the following formulas:

$$distance\ \rho = \sqrt{x^2 + y^2 + z^2} \tag{8}$$

$$azimuth\ \varphi = \arctan\left(\frac{z}{x}\right) \tag{9}$$

$$elevation\ \theta = \left(\frac{y}{\sqrt{x^2 + y^2 + z^2}}\right) \tag{10}$$

$$R = \begin{Vmatrix} \cos(R_y)\cos(R_z) & \cos(R_x)\sin(R_z) + \sin(R_x)\sin(R_y)\cos(R_z) & \sin(R_x)\sin(R_z) - \cos(R_x)\sin(R_y)\sin(R_z) \\ -\cos(R_y)\sin(R_z) & \cos(R_x)\cos(R_z) - \sin(R_x)\sin(R_y)\sin(R_z) & \sin(R_x)\cos(R_z) + \cos(R_x)\sin(R_y)\sin(R_z) \\ \sin(R_y) & -\sin(R_x)\cos(R_y) & \cos(R_x)\cos(R_y) \end{Vmatrix} \quad (4)$$

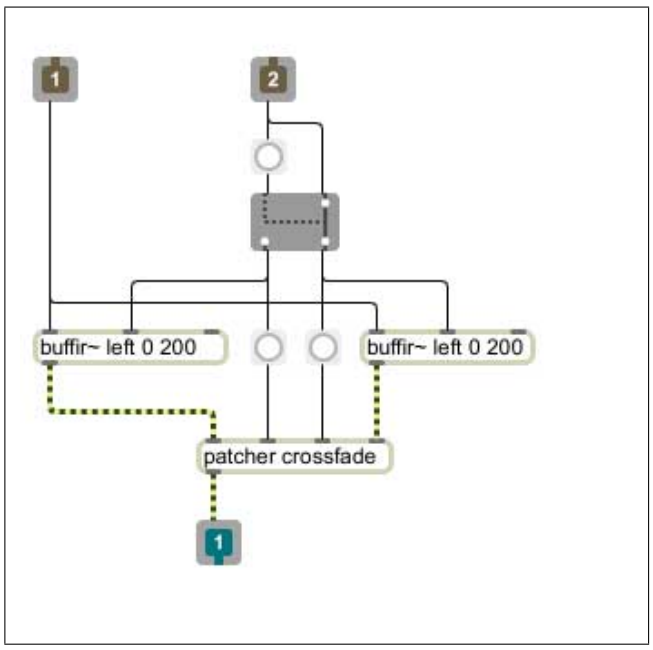


Figure 4. The detail of the MAX subpatch for the convolution process.

The new set of coordinates can be employed to retrieve the right HRIR from the database. Since our database includes only HRIRs measured at a given distance, we only use azimuth and elevation. How to use the distance value to simulate the perception of distance will be explained in Section 5.4. Since not all the possible pairs of azimuth and elevation have a corresponding measured HRIR within the database, we choose the database candidate that minimizes the euclidean distance.

5.2 The Convolution Process

This section describes the convolution process between an anechoic signal and a binaural HRIR. We use the CIPIC database [11], consisting of a set of responses measured for 45 subjects at 25 different values for azimuth and 50 different values for elevation. Each impulse consists of 200 samples.

Figure 4 illustrates the detail of the subpatch for one channel. From its first inlet it receives the anechoic signal, while from the second it gets the index for HRIR within a *buffer*$\sim$ object. HRIRs are stored in a single file that concatenates all the impulses. Please note that the process is performed one time for left channel and another one for right channel. Inside the database, azimuth and elevation values are numbered through an ad hoc mapping. Given an azimuth position naz and an elevation position nel we can calculate the starting point within the buffer with the

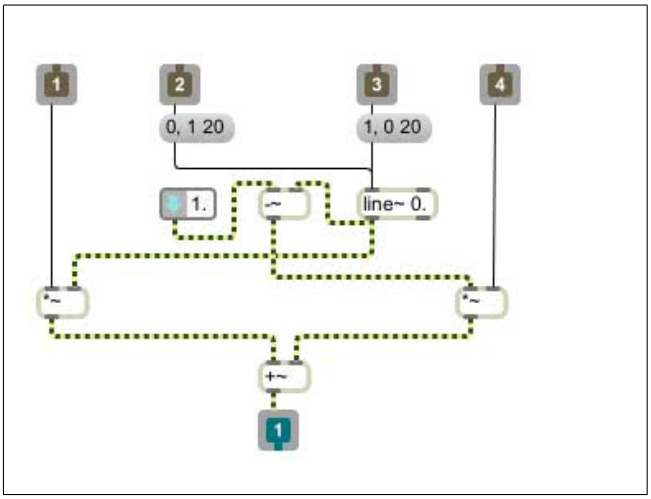


Figure 5. The detail of the MAX subpatch for the crossfade system.

formula:

$$[((naz - 1) \cdot 50) + (nel - 1)] \cdot ir_{length} \quad (11)$$

A *buffir*$\sim$ object is a finite impulse response (FIR) filter that loads both coefficients from the buffer and audio signal, and then performs convolution in time domain. Convolution is implemented through a FIR filter since the small number of samples of HRIRs makes computationally convenient to perform it in time domain instead of frequency domain. *buffir*$\sim$ object allows to store up to 256 coefficients.

5.3 Interpolation and Crossfade Among Samples

One of the known problems related to the use of HRIR for spatialization is the interpolation between two signals convolved with two different impulses. This is a very common case for this kind of real-time applications because when moving from one azimuth value to another impulses are very dissimilar. As a consequence, output signal can change abruptly, thus affecting negatively the perceived quality of the system. We have designed a simple yet performing interpolation procedure based on crossfade to limit the artifacts produced by the switch between impulses.

The approach is replicating the audio stream for each channel that lead to changes to the convolution subpatch. We add a second *buffir*$\sim$ object so now the first filter will produce signals convolved with the current impulse and the second filter will be loaded with the new HRIR provided by the new position. Then new signal will gradually overcome the signal from other filter with a crossfade function. Once done the role of the two filter will switch. This behaviour is achieved trough a *ggate*$\sim$ object.

As a performance issue it should be noted that in a real time environment every redundant operation should be avoided. In our implementation this means that a crossfade between samples is needed only if a switch has been detected by a change object that gives a value in output only if it is not equal to its previous value. This avoid unnecessary computation by the CPU that are useless if applied to the same impulse response and could lead to a degradation in terms of quality. Another improvement is given by the use of *speedlim*$\sim$ object that establish the frequency of messages in terms of minimum number of milliseconds between each consecutive message. It could happen that changing azimuth and elevation at the same time two different new messages could be generated in a rapid sequence. That could lead to a premature refresh in the filter coefficients leading to a loss of quality. With this component they are spaced by at least 40 msec. This value is chosen according with the typical refresh rate of a video stream (25 fps). This value is also used to define the crossfade duration between samples, and in our implementation the crossfade is linear. The user can define a value between 5 msec and 20 msec. By experiments, depending on the CPU power, it is possible to achieve a good quality even at 5 msec. So the overall delay between changes is $20\ msec + \frac{200_{samples}}{44100\frac{samples}{sec}}$.

5.4 Simulation of Distance

One of the limitations of the CIPIC database is presenting candidates only at one given distance. In order to simulate the distance effect, our patch contains a simple procedure based on the inverse square law. The function is implemented by an *expr*$\sim$ object [3] with the expression:

$$20\ log_{10}\left(\frac{1}{distance}\right)\ dB \qquad (12)$$

We limit the range of the distance value produced by the head-tracking system between 0.1 and 2. Conventionally 1 identifies the reference distance of the impulse response, and in this case no gain is applied. The mentioned distance value is employed to feed the gain of each channel. The process could be enhanced by adding a filter which simulates the air absorption or using a database where HRIRs are measured at various distances.

5.5 The Graphical User Interface

The software application that implements the algorithms described before is a standard patch for MAX/MSP. The patch uses an ad hoc external to implement the head-tracking function.

After launching it, the software presents a main window made of a number of panels and a floating window containing the image coming from the webcam after faceAPI processing. In the latter window, when a face is recognized, a wireframe contour is superimposed over the face image.

In Figure 6 we present the user interface of the application. As regards the main window, it is organized in several

[3] An *expr*$\sim$ object evaluates C-like expressions.

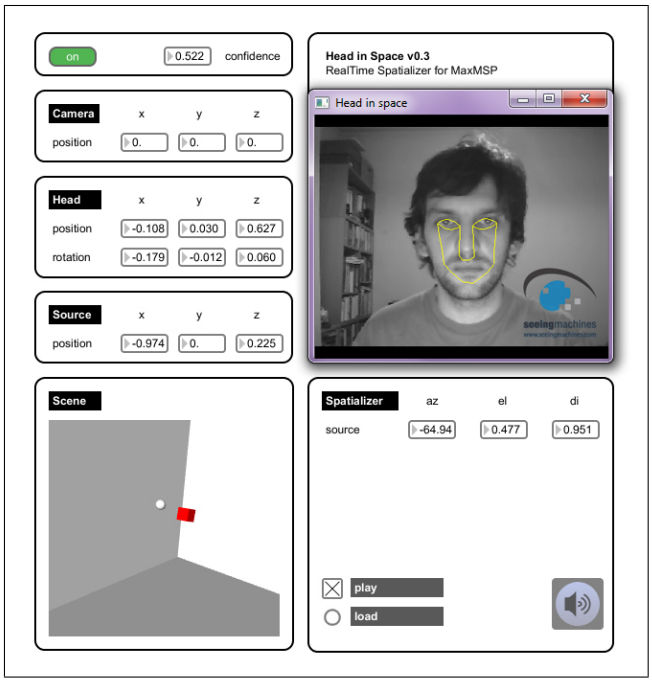


Figure 6. The graphical user interface of the program.

panels. First, it allows to switch on and off the processing engine. Besides, a number of text boxes and buttons are used to set the position of the camera and of the source. Other controls give feedback about the derived position of the listener and the corresponding translation into azimuth, elevation, and distance. A 3D representation (with the use of the OpenGL support of Jitter) of the system made of the listener (dark cube) and the source (white sphere) is also provided and updated in real time.

The bottom right panel contains the controls to choose the audio file to be played and to start the playback.

6. CONCLUSIONS & FUTURE WORKS

This paper has described a working application that performs real-time spatialization of an audio signal based on the position of the listener.

The system can be improved in several manners. The use of a single webcam corresponds to a limited resolution of azimuths and elevations ($\pm$ 90 azimuth, -30/+60 elevation, data coming from faceAPI specifications). It could be possible to combine more cameras in order to fully represent the space choosing the one with the highest confidence value.

Another improvement is adding support for more than one source in order to render a richer environment. It could also be interesting to take into account moving sound sources; this implies that a way to describe trajectories needs to be implemented.

The use of CIPIC database limits the number of possible measured distances and led us to implement a distance simulation mechanism, whereas it would be desirable to switch among HRIRs measured at various distances. Also the 200-samples HRIRs do not account for rooms ambience, so a reverberation tool is needed.

Since the application is structured in modules, it can be easily extended in order to support the future changes we have mentioned.

The source code and application are freely available from the authors at:
`http://www.lim.dico.unimi.it/HiS`.

7. REFERENCES

[1] J. Steuer, "Defining virtual reality: Dimensions determining telepresence," *Journal of Communication*, vol. 42, pp. 73–93, 1992.

[2] K. Osberg, *But what's behind door number 4? Ethics and virtual reality: A discussion*. Human Interface Technology Lab Technical Report R-97-16, 1997.

[3] S. P. Parker, G. Eberle, R. L. Martin, and K. I. McAnally, "Construction of 3-d audio systems: Background, research and general requirements.," tech. rep., Victoria: Defence Science and Technology Organisation, 2000.

[4] D. R. Begault, *3-D sound for virtual reality and multimedia*. Cambridge, MA: Academic press Professional, 1994.

[5] V. Choqueuse, "Binaural spatializer (for maxmsp)," *http://vincent.choqueuse.free.fr/*, 2007.

[6] J. Jot and O. Warusfel, "Spat˜: A spatial processor for musicians and sound engineers," in *CIARM: International Conference on Acoustics and Musical Research*, 1995.

[7] M. Noisternig, T. Musil, A. Sontacchi, and R. Hoeldrich, "3d binaural sound reproduction using a virtual ambisonics approach," in *VECIMS - International Symposion on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, (Lugano, Switzerland), 2003.

[8] T. Tatli, "3d panner: A compositional tool for binaural sound synthesis," *International Computer Music Conference*, 2009.

[9] W. A. Yost, *Foundamentals of hearing: An introduction*. Academic press London, third ed., 1994.

[10] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization*. Cambridge, MA: MIT Press, revised ed., 1996.

[11] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avedano, "The cipic hrtf database," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. W2001–1/W2001–4, October 2001.

[12] "Seeing machines face tracking api documentation," *http://www.seeingmachines.com/product/faceapi/*, 2009.

[13] A. Cipriani and M. Giri, *Musica Elettronica e Sound Design*, vol. 1. ConTempoNet, 2009.

[14] D. Zicarelli, J. Clayton, and R. Sussman, *Writing External Objects for Max and MSP 4.3*. 2001.

A LOOK INTO THE PAST: ANALYSIS OF TRENDS AND TOPICS IN PROCEEDINGS OF SOUND AND MUSIC COMPUTING CONFERENCE

Pratyush[1] **Martí Umbert**[2] **Xavier Serra**[3]

Music Technology Group,
Universitat Pompeu Fabra, Barcelona.

[1]akpratyush@gmail.com
[2] marti.umbert@gmail.com
[3] xavier.serra@upf.edu

ABSTRACT

In this paper we analyze the proceedings of all the past six editions of the *Sound & Music Computing Conference*. The proceedings are analyzed using knowledge based "keywords to text" mapping to discover the overall conference trends. The analysis is done on the basis of number of papers, distinct authors, participation ratio for each relevant topic, interdependence of topics in terms of shared keywords and the overall popularity of keywords. The analysis was done for each conference year as well as for the overall collection of proceedings till date. The objective of the discussed work is to provide an insight over the past six years in the SMC community that was envisioned in the roadmap.

1. INTRODUCTION

Since its first conference in 2004 to Porto's 2009 conference, the Sound & Music Computing field has traveled a path, which has given it a status that it could be treated as a standalone scientific discipline. The aim of this paper is to analyze the evolution of the publications of the SMC conference that was envisioned in the roadmap of the SMC field [1]. Since these publications were scattered over the Internet in individual websites of each conference year, it was not easy to extract information about the conference.

For the ease of data retrieval & analysis of the publications on the basis of author participation, topic of interests, relationship between different topics & the trend over the years we decided to build a web repository of all the publications of the SMC Conference till date and used it for the analysis reported in this particular paper. Contrary to the other works presented in [2, 3], the current work relies on collaborative effort, knowledge simplification and rule based classification.

Section 2 describes how this repository was built as well as its preliminary analysis. The methodology used for the analysis of the papers is explained in section 3 while section 4 presents the results of the analyses that were performed and finally in section 5 we discuss the conclusion of the analyses and the work done.

2. THE REPOSITORY

2.1 Building the repository

Every published paper of the SMC Conference so far was downloaded from each year's conference's individual website and then manually entered in a relational database management system. Drupal Content Management Service [4] was chosen as the framework.

2.2 Preliminary Analysis

Preliminary analysis of the built repository was done in order to expose the trend about number of papers & author participation in each edition as well in the overall history of the conference. The results of the preliminary analysis are summarized in Table 1.

Year	No. of Papers	No. of Distinct Authors
2004	46	83
2005	31	68
2006	25	50
2007	60	116
2008	34	70
2009	62	163
Overall	**258**	**482**

Table 1. Preliminary analysis of the SMC Publications.

It has to be taken into account that the overall number of distinct authors (482) is not the sum of the individual distribution of distinct authors for each conference year as an author is most likely to participate in two or more editions of the conference. Specifically, over the years, 68 authors have participated in more than one edition of the conference.

High level topics	Middle level topics	
	Name	**ID**
Processing of sound and music signals	3D sound/music, Sound/music signal processing algorithms, Digital Audio Effects, Musical sound source separation	Topic 1
	Sound synthesis, Spectral modeling synthesis, Physical modeling for sound generation	Topic 2
Understanding and modeling sound and music	Music information retrieval, Musical pattern recognition/modeling, Computational musicology, Technologies for the preservation, access and modeling of musical heritage, Automatic music transcription, Musical sound source separation and recognition	Topic 3
	Music and emotions, Sound/music and Neuroscience, psychology, psychoacoustics, Sound/music perception and cognition	Topic 4
Interfaces for sound and music	Interfaces for music creation and fruition, Gesture controlled audio systems, Mobile music, Interactive performance systems, Musical performance modeling Visualization of sound/music data, Sonic interaction design	Topic 5
Assisted sound and music creation	Web 2.0 and music, Networked music generation	Topic 6
	Computer environments for sound/music processing, Automatic music	Topic 7

Table 2. Hierarchy between High & Middle Level Topics

3. THE METHODOLOGY

This section describes the overall process, which was followed in order to get the analysis result that was envisioned. The process starts with simplification of knowledge to generate "List of Topics" associated with the Sound & Music Computing community, secondly a handful of "Keywords" were created to map each paper to one or the other topics in the list. Finally, statistical analyses of the papers were done.

3.1 Knowledge simplification

To classify all the papers of the previous SMC Conferences on the basis of "Research Topics", we started with the topics for call for papers for the upcoming Sound & Music Conference. Since these topics were very specific in nature, we classified them into broader topics. The classification, of the original topics into broader ones was done on the basis of "Similarity in Concepts".

This led to the construction of a hierarchical classification, constituting two levels of simplification:

1. "Middle Level Topics"
2. "High Level Topics"

The mappings of these two levels of topics are presented in Table 2.

3.2 Keyword building

Since the papers presented in the SMC Conference did not have "Keywords" in the full text, there were two options for building a keyword repository for the papers:

1. Automatic extraction of keywords from the abstract of the papers using a probabilistic mixture model as introduced in [5].

2. Use of CSCW methods using Google docs as discussed in [6] to collaborate with other researchers of the field to have a consensus on a set of keyword for each topic.

For the current work, we used the second method for building the keyword list for each middle level topic. The consensus was reached using cross validation of the keywords between each researcher in the second pass of the questionnaires. Furthermore, relevance of each keyword was checked by searching for the keyword in the SMC papers as well as searching for papers using the same keyword in Google scholar.

3.3 Search Mechanism

The search mechanism attempts to assign a topic to each paper, based on the keyword. The entire process can be described as below:

1. Search every keywords of each topic in the abstract & title of each paper.
2. If a particular keyword is present in the abstract or title, we add that keyword & the associated topic as a contender for classifying that paper.
3. Once the keywords are mapped in a particular paper, we count the total number of occurrences of each keyword & the total presence of keywords from each topic.

4. The topic with the maximum number of keywords present in a paper is decided to be the relevant topic for that paper.
5. If more than one topic has equal presence in a paper, we classify the paper as *'Multiple Topic'*.
6. If none of the keywords could be mapped to a paper, we label the paper as *'Unknown Topic'*.

This process is carried out to classify a paper to a Middle Level topic using the set of corresponding keywords for each topic. For the re-classification of each paper based on High Level Topics, we use the relationship between the Middle Level & High Level topics as depicted in Table 2. Furthermore, if there were discrepancies in the high level classification, we assigned those papers as 'Unclassified'.

Topic	**'04**	**'05**	**'06**	**'07**	**'08**	**'09**	**ALL**
Multiple	**13**	4	5	15	8	11	56
No topic	4	0	1	2	3	3	13
Topic 1	3	2	1	3	7	3	19
Topic 2	9	**13**	**7**	13	**9**	10	**61**
Topic 3	7	0	5	4	1	13	30
Topic 4	2	0	1	0	0	2	5
Topic 5	5	11	4	**18**	3	**15**	56
Topic 6	3	1	1	4	3	5	17
Topic 7	0	0	0	1	0	0	1

Table 3. Year wise distribution showing the absolute number of papers for each Middle Level Topic.

Topic	**'04**	**'05**	**'06**	**'07**	**'08**	**'09**	**ALL**
Unclassified	**13**	4	5	13	6	10	51
No topic	4	0	1	2	3	3	13
Assisted…	3	1	1	5	3	5	18
Interfaces…	5	11	4	**18**	3	**15**	56
Processing…	12	**15**	**8**	**18**	**17**	14	**84**
Understanding…	9	0	6	4	2	**15**	36

Table 4. Year wise distribution showing the absolute number of papers for each High Level Topic.

4. RESULTS

The results that we obtained after the analysis are presented in this section. For better aesthetics of the plots & charts, we have used aliases for each Middle Level Topic.

4.1 Participation ratio for each relevant topic

Since each paper was classified as either a topic or multiple topic or unknown topic, we can deduce the distribution of each topic in each year's conference as well as in the overall conference till date.

Figure 1 shows the distribution of each Middle Level Topic in the overall conference history, while Table 3 is used to visualize the distribution of each Middle Level Topic in each edition of the conference.

Likewise, Figure 2 displays the distribution of the High Level Topics in all years of the conference taken together, whereas Table 4 is used to show the distribution of these topics in each conference year.

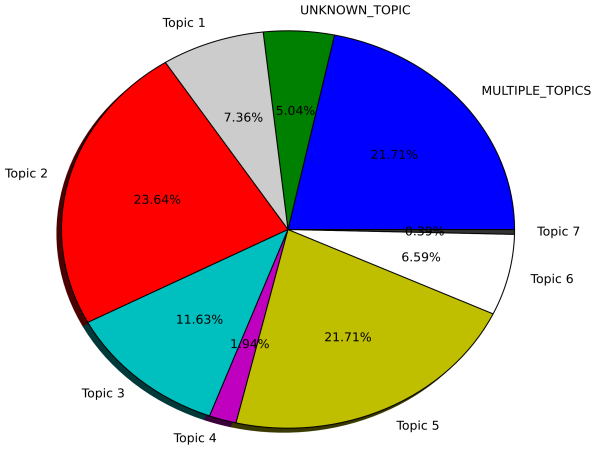


Figure 1. Publication distribution for Middle Level Topics for the overall conference till date.

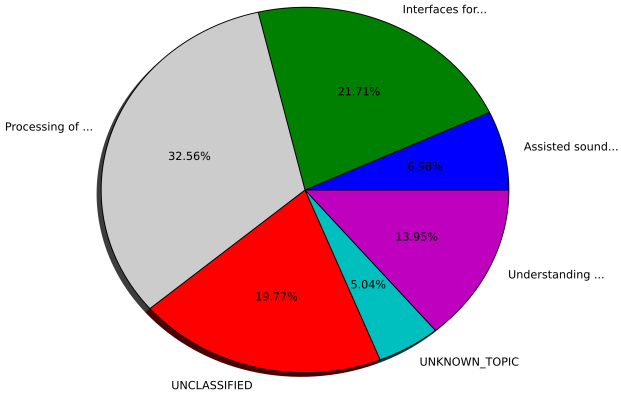


Figure 2. Publication distribution for High Level topics for all years taken together.

4.2 Trends for each level of topics over the entire conference history

The change in the number of papers for each topic over the years is presented both for Middle Level & High Level topics in Figure 3 and Figure 4 respectively.

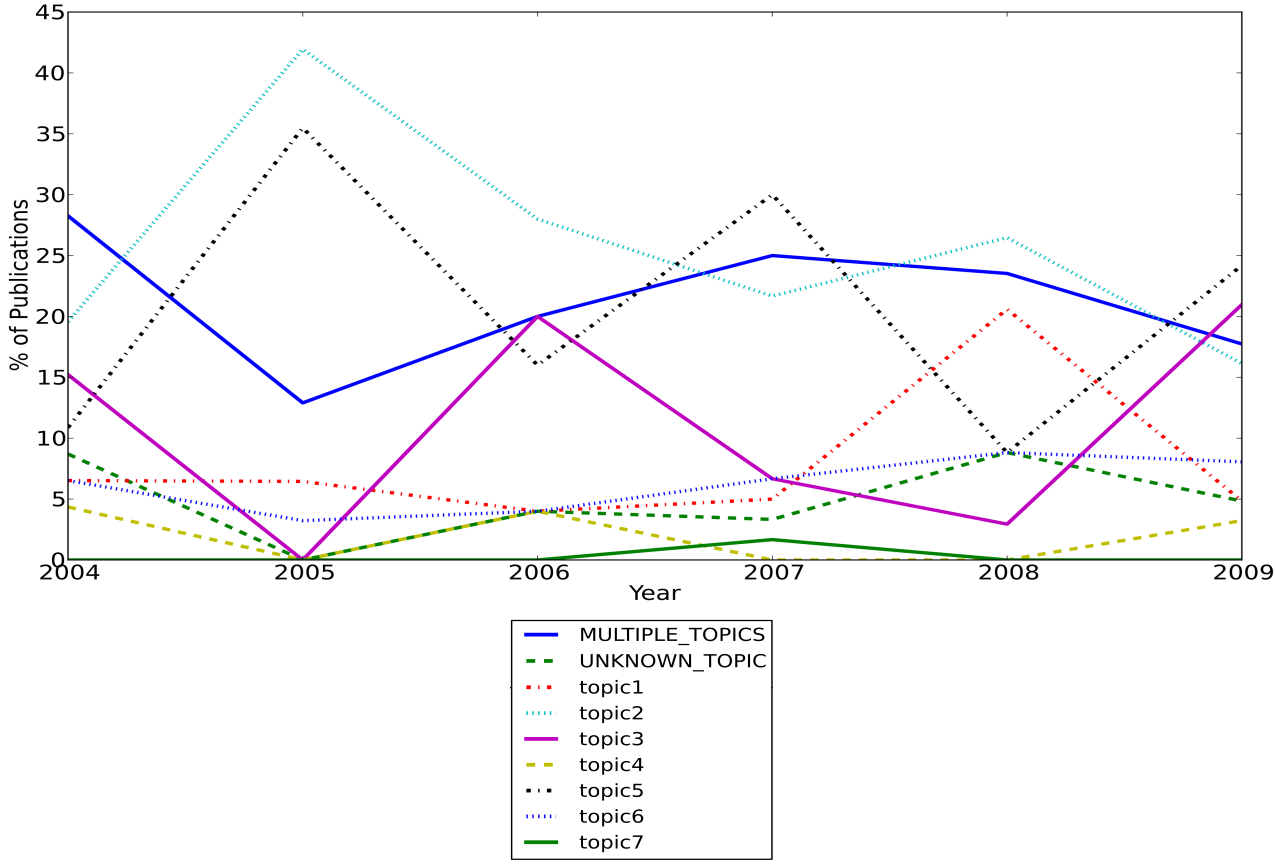


Figure 3. Middle Level Topic trend.

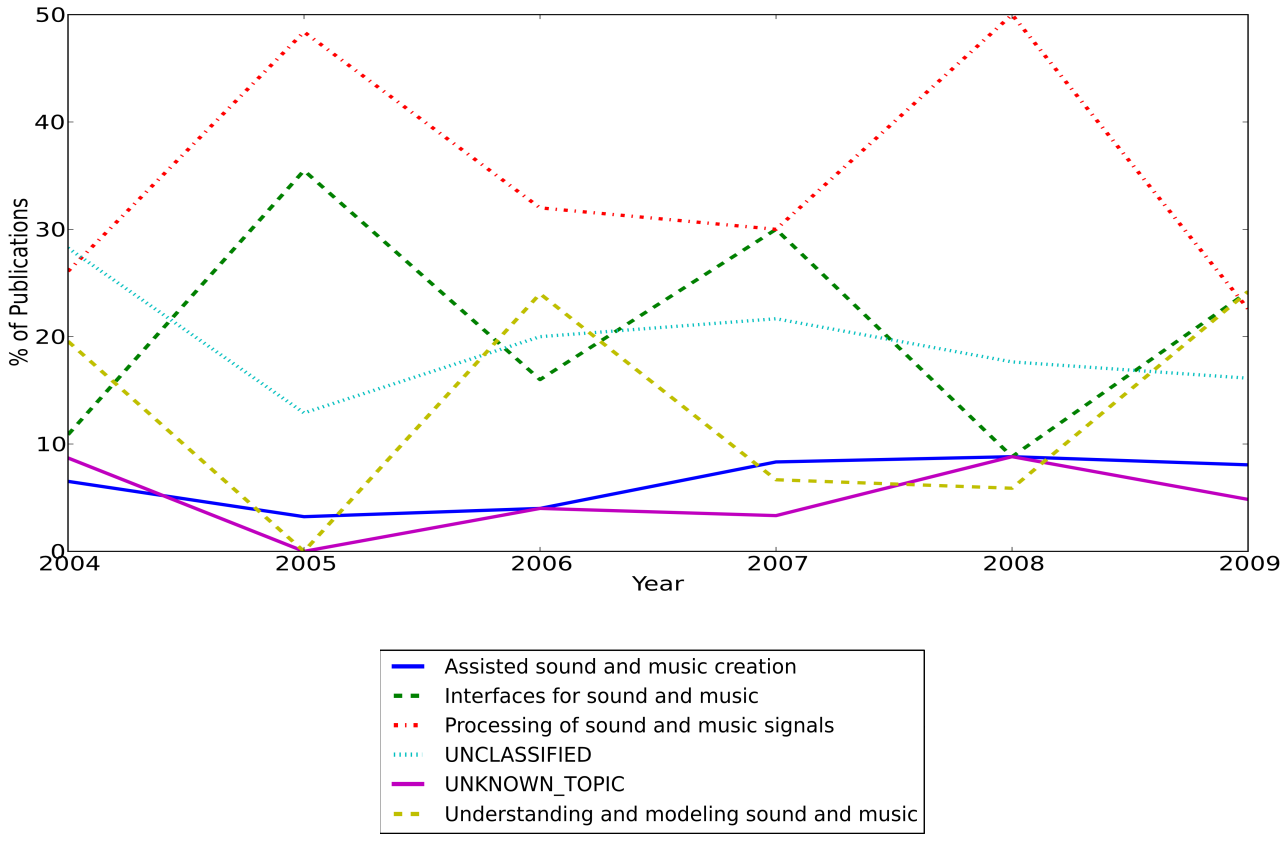


Figure 4. High Level Topic trend.

4.3 Closeness between the topics

Since we used decision based approach for assigning a paper a relevant topic based on the "presence of keyword", we observed many papers which were classified as particular topic but had a fair amount of keywords of other topics were present as well. This can be used to deduce the closeness of a topic with others. The cross infiltration (presence) of each Middle Level topic in every other for the overall conference publications is showed in Table 5.

Topic	1	2	3	4	5	6	7
Topic 1	*58.55*	*18.42*	3.95	0.66	13.82	4.61	0.00
Topic 2	9.37	*59.10*	7.21	1.26	*17.66*	5.23	0.18
Topic 3	6.48	10.65	*60.19*	4.63	*16.20*	1.85	0.00
Topic 4	12.20	12.20	12.20	*43.90*	*17.07*	2.44	0.00
Topic 5	15.43	9.88	3.09	2.78	*51.23*	*17.44*	0.15
Topic 6	2.26	15.04	2.26	3.76	*24.06*	*52.63*	0.00
Topic 7	0.00	0.00	0.00	0.00	0.00	0.00	*100.00*

Table 5. Presence of each topic in each other (Middle Level).

4.4 Keywords & their relevance

As the keywords play a pivotal role in the overall procedure that we presented here, we found out the popularity of each individual keyword irrespective of the topic they represent in all the papers published in the SMC Conference till date.

A keyword cloud representing the popularity or presence of these keywords is plotted below as Figure 5. The most frequent 50 keywords are shown with a font size that reflects this popularity. Frequency values range from 6 to 119 occurrences.

Figure 5. Keyword Cloud

5. CONCLUSION

In this paper, we analyzed the proceedings of the past SMC Conferences, tried to categorize each published paper into one of the proposed 7 Middle Level & 4 High Level Topics so that the trend of the SMC Conferences could be identified and justified.

To start with we noticed that 482 authors have participated in the SMC Conferences till 2009 and out of those, 68 authors have publications in more than one edition of the conference.

For e.g. we found out that *Topic 2: Sound synthesis, Spectral modeling synthesis, Physical modeling for sound generation* and *Topic 5: Interfaces for music creation and fruition, Gesture controlled audio systems, Mobile music, Interactive performance systems, Musical performance modeling Visualization of sound/music data, Sonic interaction design* remains the most popular topic throughout the conference with a combined share of ~77% in SMC Conference 2005 and ~45% overall. Of all the conferences till date, the share of *Topic 1: 3D sound/music, Sound/music signal processing algorithms, Digital Audio Effects, Musical sound source separation* was highest in 2008 about 20% and *Topic 3: MIR & others* had a considerable share in the 2009's conference with about 21% publications.

From the participation ratio of each Middle Level topic in each year, we find the following trends in the evolution of some topics over the years:

1. Web 2.0 grows since 2005, this can be justified by the fact that web 2.0 evolved a lot since that time, so it attracted much research in the recent years.
2. Sound synthesis/ signal processing has a slight decline in percentage in the recent years this might be because the growing popularity of other fields.
3. Since the theme of the 2008 conference was "Sound in Space", the abrupt increase in the number of publications of the topic "3D Audio" for that year is justified.

From the closeness analysis of each topic Vs the others, we could clearly see that Topic 1 & Topic 2 are closely related to each other, so our classification of grouping them together in the higher level of classification is fairly justified. Although we have grouped Topic 6 & Topic 7 together, this is not fairly justified by the data presented in Table 5. This is due to the fact that there is a hairline difference between the last two high level topics and thus Topic 5 & Topic 6 are also closely related as depicted in the same table. Alternatively, Topic 5, 6 & 7 could be regrouped to a new High Level topic as well.

And finally looking at the Keyword cloud, we could see that the popular keywords from the set we had, are synthesis, analysis, instrument, realtime, voice, net, etc.

Since the overall methodology relied on text mining and knowledge simplification using which we classified a dataset of nearly 31000 words with a keyword set of 117 keywords, the evaluation of the system is tough. Moreover, the evaluation is also hampered by the fact that there were no keywords provided by the authors in the SMC papers to cross check with.

To conclude with, we would like to highlight that only 5% of the papers were of *unknown topic* and about 21% of the papers were of Multiple Topics (*unclassified*), this correlates to the fact that Sound & Music Computing is highly inter-disciplinary in nature. Another point to take into account is that these conclusions have been deduced from the last 6 SMC proceedings, which might not represent enough data to support them.

Also, we would like to continue to explore the presence of research groups of different universities in the SMC Conferences based on publications and how papers, authors & research topics could be classified together on the basis of co-authorship, citations and bibliographic links.

6. REFERENCES

[1] A Roadmap for Sound and Music Computing. Version 1.0. , 2007. Available at: http://www.smcnetwork.org/roadmap

[2] J.H. Lee, J.S. Downie and M.C. Jones: "An analysis of ISMIR proceedings: Patterns of authorship, topic, and citation", *10th International Society for Music Information Retrieval Conference,* pp. 57-62, 2009

[3] G. Widmer et al, : "The ISMIR Cloud: A decade of ISMIR Conferences at your fingertips", *10th International Society for Music Information Retrieval Conference,* pp. 63-68, 2009

[4] Official drupal website available at http://drupal.org

[5] A. Velivelli, B.Yu and C. Zhai: "A cross-collection mixture for comparative text mining," *Proceedings of ACM-KDD*, pp. 743–748, 2004.

[6] S. Dekeyser and R. Watson: "Extending Google Docs to Collaborate on Research Papers". *Technical report, The University of Southern Queensland, Australia,* 2006.

MUSICGALAXY – AN ADAPTIVE USER-INTERFACE FOR EXPLORATORY MUSIC RETRIEVAL

Sebastian Stober and Andreas Nürnberger
Data & Knowledge Engineering Group, Faculty of Computer Science
Otto-von-Guericke-University Magdeburg, Germany
{sebastian.stober, andreas.nuernberger}@ovgu.de

ABSTRACT

Sometimes users of a music retrieval system are not able to explicitly state what they are looking for. They rather want to browse a collection in order to get an overview and to discover interesting content. A common approach for browsing a collection relies on a similarity-preserving projection of objects (tracks, albums or artists) onto the (typically two-dimensional) display space. Inevitably, this implicates the use of dimension reduction techniques that cannot always preserve neighborhood and thus introduce distortions of the similarity space. This paper describes ongoing work on MusicGalaxy – an interactive user-interface based on an adaptive non-linear multi-focus zoom lens that alleviates the impact of projection distortions. Furthermore, the interface allows manipulation of the neighborhoods as well as the projection by weighting different facets of music similarity. This way the visualization can be adapted to the user's way of exploring the collection. Apart from the current interface prototype, findings from early evaluations are presented.

1. INTRODUCTION

There is a lot of ongoing research in the field of music retrieval aiming to improve retrieval results for queries posed as text, sung, hummed or by example as well as to automatically tag and categorize songs. All these efforts facilitate scenarios where the user is able to somehow formulate a query – either by describing the song or by giving examples. But what if the user cannot pose a query because the search goal is not clearly defined? E.g., he might look for background music for a photo slide show but does not know where to start. All he knows is that he can tell if it is the right music the moment he hears it. In such a case, exploratory retrieval systems can help by providing an overview of the collection and letting the user decide which regions to explore further.

When it comes to get an overview of a music collection, neighborhood-preserving projection techniques have become increasingly popular. Beforehand, the objects to be projected – depending on the approach, this may be artists, albums, tracks or any combination thereof – are analyzed to extract a set of descriptive features. (Alternatively, feature information may also be annotated manually or collected from external sources such as the EchoNest API [1].) Based on these features, the objects can be compared – or more specifically: appropriate distance- or similarity measures can be defined. The general objective of the projection can then be paraphrased as follows: Arrange the objects in two or three dimensions (on the display) in such a way that neighboring objects are very similar and the similarity decreases with increasing object distance (on the display). Popular dimensionality reduction techniques for such a neighborhood-preserving projection are self-organizing maps (SOM) [1], principal component analysis (PCA) [2] and multidimensional scaling techniques (MDS) [3]. As the feature space of the objects to be projected usually has far more dimensions than the display space, the projection inevitably causes some loss of information – irrespective of which dimensionality reduction techniques is applied. Consequently, this leads to a distorted display of the neighborhoods such that some objects will appear closer than they actually are, and on the other hand some objects that are distant in the projection may in fact be neighbors in feature space.

Another problem that arises when working with similarity-based neighborhoods is that music similarity is highly subjective and may depend on a person's background. Consequently, there is more than one way to look at a music collection – or more specifically to compare two tracks based on their features: A musician, for instance, might especially look after structures, harmonics or instrumentation (possibly paying – conscious- or unconsciously – special attention to his own instrument). Non-musicians will perhaps focus more on overall timbre or general mood. Others, in turn, may have a high interest in the lyrics as long as they are able to understand the particular language. A music retrieval system should be able to incorporate this subjectiveness in order to better comply with the individual needs of its users and consequently gain a higher acceptance. To this end, the system presented in this paper allows the user to modify the underlying distance measure by adapting weights for different aspects of similarity. Further, the described user-interface exploits the above mentioned distorted neighborhood relations during user-interaction. The approach is based on a multi-focus fish-eye lens that allows a user to enlarge and explore a region

[1] http://developer.echonest.com/

of interest while at the same time adaptively distorting the remaining collection to reveal distant regions with similar tracks.

A broad overview of related work and a discussion of computational complexity aspects has been given in [4]. This paper focuses primarily on the user-interface and the interaction. To this end, the fundamental preprocessing steps are reviewed in Section 2. Section 3 covers important aspects of the visualization with Section 3.2 focusing on the SpringLens distortion technique in particular as the basis for the user-interaction. The user-interaction is described in Section 4. Specifically, Section 4.2 provides insight into how the neighborhood distortions caused by the projection are addressed. The evaluation process of the user-interface is described by Section 5. Finally, Section 6 concludes the paper.

2. DATA PREPROCESSING

This section sketches all necessary preprocessing steps that can be done offline, i.e. before the actual interaction with the user. For a detailed description, we refer to [4].

2.1 Feature Extraction

The prototype system described here uses collections of music tracks. As a prerequisite, it is assumed that the tracks need to be represented by some descriptive features that can, e.g., be extracted, manually annotated or obtained form external sources. In the current implementation, content-based features are extracted utilizing the capabilities of the frameworks CoMIRVA [5] and JAudio [6]. Specifically, Gaussian Mixture Models of the Mel Frequency Cepstral Coefficients (MFCCs) according to [7] and [8] and "fluctuation patterns" describing how strong and fast beats are played within specific frequency bands [9] are computed with CoMIRVA. JAudio is used to extract a global audio descriptor "MARSYAS07" as described in [10]. Further, lyrics for all songs were obtained through the web service of LyricWiki [2], filtered for stop words, stemmed and described by document vectors with TFxIDF term weights [11]. Additional features that are currently only used for the visualization are ID3 tags (artist, album, title, track number and year) extracted from the audio files, track play counts obtained from a last.fm profile, and album covers gathered through web search.

2.2 Facet Definition

Based on the features associated with the tracks, *facets* are defined that refer to different aspects of music (dis-) similarity:

Definition Given a set of features F, let S be the space determined by the feature values for a set of objects O. A *facet distance metric* d (or short: *facet*) is a distance metric defined on a subspace $S' \subseteq S$ of the feature space and satisfying the following conditions for any $x, y, z \in O$:

- $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$ (symmetry)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

For instance a facet "timbre" could be defined on the MFCC-based feature described in [8] whereas a facet "text" could compare the combined information from the features "title" and "lyrics". [3]

In order to be able to aggregate values from several facet distance metrics, the following normalization is applied for all distance values v of a facet d:

$$v' = \min\{1, \frac{v}{\mu + \sigma}\}$$

where μ is the mean and σ the standard deviation of all distance values with respect to d. This truncates very high distance values and results in a value range of $[0, 1]$. For the aggregation, basically any function could be used. In the current prototype the following parametrized aggregation functions are predefined:

- $d = \sqrt{\sum_{i=1}^{l} w_i d_i^2}$ (weighted euclidean distance)
- $d = \sum_{i=1}^{l} w_i d_i^2$ (squared weighted eucl. distance)
- $d = \sum_{i=1}^{l} w_i d_i$ (weighted sum)
- $d = \max_{i=1..l}\{w_i d_i\}$ (maximum)
- $d = \min_{i=1..l}\{w_i d_i\}$ (minimum)

The aggregation functions allow to control the importance of the facet distances $d_1, \ldots, d_l$ through their associated weights $w_1, \ldots, w_l$. Default settings for the facet weights and the aggregation function are defined by an expert (who also defined the facets themselves) and can later be adapted by the user during interaction with the interface. Table 1 lists the facets used in the current implementation.

facet name	feature	distance metric
timbre	GMM of MFCCs	euclidean distance
rhythm	fluctuation patterns	euclidean distance
dynamics	MARSYAS07	euclidean distance
lyrics	TFxIDF vectors	cosine distance

Table 1. Facets defined for the current implementation.

2.3 Indexing

A small sample of tracks is drawn from the collection with the objective to approximate the covariance matrix of the whole collection. These tracks are referred to as *landmarks*. Only for these tracks, the projection will be computed later. The remaining tracks will be placed according to their distances to the landmarks. This reduces the computational complexity of the projection method and facilitates near real-time updates in case of parameter changes.

[2] http://lyricwiki.org

[3] It is important to stress the difference to common faceted browsing and search approaches that rely on a faceted classification of objects to support users in exploration by filtering available information. Here, no such filtering by value is applied. Instead, we employ the concept of facet distances to express different aspects of (dis-)similarity that can be used for filtering.

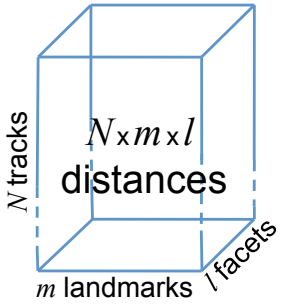


Figure 1. Facet distance cuboid data structure holding the distance values for N tracks, m landmarks and l facets.

To further speed up the projection, the distances to all landmarks are precomputed and stored in a 3-dimensional data structure called "facet distance cuboid" (Figure 1) that holds for each song the distance values to all landmarks for all facets. (As it is not clear at indexing time how the facet distance values are to be aggregated, the values need to be stored separately.) Note that the space requirement effectively scales linearly with the number of tracks as the number of landmarks and facets can be considered small or even constant in comparison so that the data structure should fit into RAM even for large collections.

Further, a neighbor index needs to be constructed to facilitate fast lookup of neighbors during the interaction. Currently, nearest neighbors are precomputed for each track and updated each time the facet aggregation parameters change. This is not appropriate for large collections and efficient indexing techniques need to be investigated in future work. Possible candidates are space partition trees [12] or approaches based on locality sensitive hashing [13] that may even be kernelized [14] to allow for more complex distance metrics. However, as our current research focus lies on the user-interface, this is only a secondary problem.

3. VISUALIZATION

3.1 Projection Technique

Before the collection can be projected, the facet distances need to be aggregated. The facet distance cuboid (c.f. Section 2.3) already holds the precomputed facet distance values. I.e., for each (track, landmark)-pair, a list of values referring to the respective facets can be looked up. The parametrized aggregation function that computes a single value from such a list can be fully controlled by the user.

In the projection step, each track is mapped from the high-dimensional feature space to a 2-D-coordinate that will later be used to infer its position when displayed on the screen. Naturally, this projection should be neighborhood-preserving such that tracks close to each other in feature space are also close in the projection. We apply a landmark- or pivot-based multidimensional scaling approach (LMDS) for the projection as described in detail in [15, 16]. This is a computationally efficient approximation to classical MDS that is feasible for application on large data sets as it scales linearly with the size of the data set. The general idea of this approach is as follows: Given the sample of landmark objects selected during preprocessing by some heuristic, an embedding into low-dimensional space is computed for these objects using classical MDS. Each remaining data object can then be located within this space according to its distances to the landmarks.

3.2 Distortion Technique

Once the 2-D-positions of all tracks are computed, the collection could already be displayed. However, an intermediate distortion step is introduced that serves as the basis for the interaction techniques described later.

The distortion technique is based on an approach originally developed to model complex nonlinear distortions of images called "SpringLens" [17]. A SpringLens consists of a mesh of mass particles and interconnecting springs that form a rectangular grid with fixed resolution. Through the springs, forces are exerted between neighboring particles affecting their motion. By changing the rest-length of selected springs, the mesh can be distorted. The deformation is calculated by a simple iterative physical simulation over time. This way, the SpringLens functions as a very flexible lens.

In the context of this work, we apply SpringLens to simulate a complex superimposition of multiple fish-eye lenses. We chose a moderate resolution with a maximum of 50 cells in each dimension for the overlay mesh which yields sufficient distortion accuracy while real-time capability is maintained. The distorted position of the projection points is obtained by barycentric coordinate transformation with respect to the particle points of the mesh. Additionally, z-values are derived from the rest-lengths that are used in the visualization.

3.3 Visualization Metaphor

The music collection is visualized as a galaxy. Each track is displayed as a star or as its album cover. The brightness and (to some extend) the hue of stars depends on a predefined importance measure. In the current implementation, this is simply the play count (imported from last.fm). However, it would as well be possible to use (user) ratings or sophisticated popularity measures. The size and the z-order (i.e. the order of objects along the z-axis) of the objects depends on their distortion z-values. Optionally, the SpringLens mesh overlay can be displayed. The visualization then resembles the space-time distortions well known from gravitational and relativistic physics.

3.4 Filtering

In order to reduce the amount of information displayed at a time, the user can choose between different filters that decide whether a track is displayed collapsed or expanded – i.e. as a star or album cover respectively. The resulting display using these filter modes are shown in Figure 2. Apart from collapsing or expanding all tracks, it is possible to expand only those tracks in magnified regions (i.e. with a z-level above a predefined threshold) or to apply a *sparser filter*.

A sparser filter selects only a subset of the collection to be expanded that is both, sparse (well distributed) and representative. Representative tracks are those with a high importance – a feature (currently the play count) that also influences a star's brightness and hue.

The first sparser version used a Delaunay triangulation constructed incrementally top-down starting with one big triangle. If the size of a triangle exceeds this threshold, the most important track within this triangle is chosen for display and added as a point for the triangulation. This process continues recursively until no triangle that exceeds a minimum size threshold contains anymore tracks that could be added.

The currently used sparser employs a different strategy: It divides the display into a grid of quadratic cells. The size of the cells depends on the screen resolution and the minimal display size of the album covers. Further, it maintains a list of the tracks ranked by importance that is precomputed and only needs to be updated when the importance values change. On an update, the sparser runs through its ranked list. For each track it determines the respective grid cell. If the cell and the surrounding cells are empty, the track is expanded and its cell blocked. (Checking surrounding cells avoids image overlap. The necessary radius for the surrounding can be derived from the cell and cover sizes.) This sparser approach produces more appealing results in terms of the spatial distribution of displayed covers.

Originally, the set of expanded tracks was updated after any position changes caused by the distortion overlay. However, this was considered irritating during early user tests and the sparser strategy was changed to update only if the projection or the displayed region changes.

4. INTERACTION

The user-interface as shown in Figure 3 allows several ways of interacting with the visualization [4]: Users can explore the collection through common panning & zooming (Section 4.1). Alternatively, they can use the adaptive multi-focus technique introduced with this prototype (Section 4.2). Further, they can change the facet aggregation function parameters and this way adapt the view on the collection according to their preferences (Section 4.3). Hovering over a track displays its title and a double-click start the playback that can be controlled by the player widget at the bottom of the interface. Apart from this, several display parameters can be changed such as the filtering mode (Section 3.4), the size of the displayed album covers or the visibility of the SpringLens overlay mesh.

4.1 Panning & Zooming

Panning shifts the displayed region whereas zooming decreases or increases it. These are very common interaction techniques that can e.g. be found in programs for geo-data visualization or image editing. Using the keyboard, the user can pan with the cursor keys and zoom in and out with $+$ and $-$ respectively. Alternatively, the mouse can be used: Clicking and holding the left button while moving the mouse pans the display. The mouse wheel controls the zoom level. If not the whole region can be displayed, an overview window indicating the current section is shown in the top left corner, otherwise it is hidden. Clicking into the overview window centers the display around the respective point. Further, the user can drag the section indicator around which also results in panning.

[4] A demo-video is available at http://www.dke-research.de/aucoma

4.2 Focusing

This interaction techniques allows to visualize – and to some extend alleviate – the neighborhood distortions introduced by the dimensionality reduction during the projection. The approach is based on a multi-focus fish-eye lens that is implemented using the SpringLens distortion technique (Section 3.2). It consists of a user-controlled *primary focus* and a neighborhood-driven *secondary focus*.

The primary focus is a common fish-eye lens. By moving this lens around (holding the right mouse button), the user can zoom into regions of interest. In contrast to the basic linear zooming function described in Section 4.1, this results in a nonlinear distortion of the projection. As a result, the region of interest is enlarged making more space to display details. At the same time, less interesting regions are compacted. This way, the user can inspect closely the region of interest without loosing the overview as his field of view is not narrowed (as opposed to the linear zoom). The visual effect produced by the primary zoom resembles a 2-dimensional version of the popular "cover flow" effect.

The secondary focus consist of multiple such fish-eye lenses. These lenses are smaller and cannot be controlled by the user but are automatically adapted depending on the primary focus. When the primary focus changes, the neighbor index (Section 2.3) is queried with the track closest to the center of focus. If nearest neighbors are returned that are not in the primary focus, secondary lenses are added at the respective positions. As a result, the overall distortion of the projection brings the distant nearest neighbors back closer to the focused region of interest. Figure 4 shows the primary and secondary focus with visible SpringLens mesh overlay.

As it can become very tiring to hold the right mouse button while moving the focus around, the latest prototype introduces a focus lock mode (toggled with the return key). In this mode, the user clicks once to start a focus change and a second time to freeze the focus. Further, in the previous prototype, the secondary focus was always updated instantly when the primary focus changed. In the current version, this behavior can be disabled resulting only in an update of the secondary focus once the primary focus does not change anymore.

4.3 Adapting the Aggregation Functions

Two facet control panels allow to adapt two aggregated distance metrics by choosing a function type (in a drop-down menu) and adjusting weights for the individual facets (through sliders). The first aggregated distance metric is applied to derive the track-landmark distances from the facet distance cuboid (c.f. Section 2.3). These distances

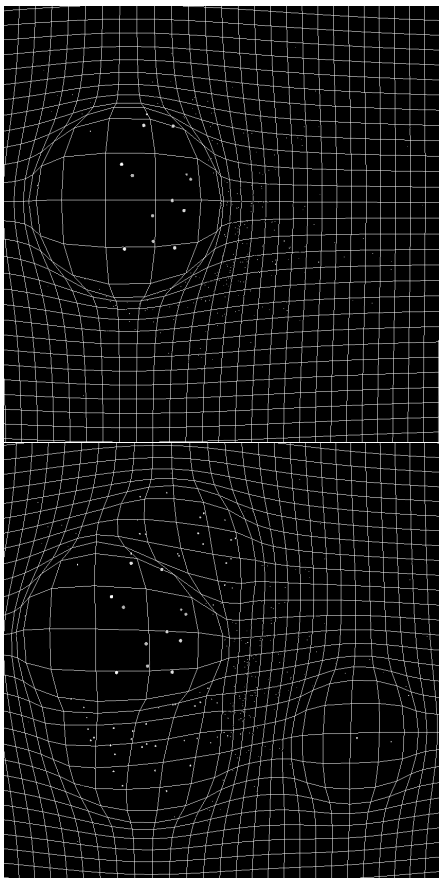


Figure 4. SpringLens distortion with only primary focus (top) and additional secondary focus (bottom).

are then used to compute the projection of the collection. The second aggregated distance metric is applied to identify the nearest neighbors of a track and thus indirectly controls the secondary focus.

Changing the aggregation parameters results in a near real-time update of the display so that the impact of the change becomes immediately visible: In case of the parameters for the nearest neighbor search, some secondary focus region may disappear while somewhere else a new one appears with tracks now considered more similar. Here, the transitions are visualized smoothly due to the underlying physical simulation of the SpringLens grid. In contrast to this, a change of the projection similarity parameters has a more drastic impact on the visualization possibly resulting in a complete re-arrangement of all tracks. This is because the LMDS projection technique produces solutions that are unique only up to translation, rotation, and reflection and thus, even a small parameter change may, e.g., flip the visualization. As this may confuse users, one direction of future research is to investigate how the position of the landmarks can be constrained during the projection to produce more gradual changes.

The two aggregated distance metrics are linked by default as it is most natural to use the same metric for projection and neighbor retrieval. However, unlinking them and using e.g. orthogonal distance metrics can lead to interesting effects: For instance, one may choose to compute the collection based solely on acoustic facets and find nearest neighbors for the secondary focus through lyrics similarity. Such a setting would help to uncover tracks with a similar topic that (most likely) sound very different.

5. EVALUATION

We follow a user-driven design approach [18] by iteratively alternating between development and evaluation phases. This paper describes the state after two significant revisions in the third development phase. The initial prototype (before the first evaluation) is described in [4] – focusing primarily on computational complexity and covering a performance evaluation of the projection and distortion methods. After some further refinements (including for instance the new sparser filter), this first prototype was presented at the CeBIT 2010 fair[5] in early March 2010. During the fair, feedback was collected from a total of 112 visitors aged between 16 and 63 years. The general reception was very positive. The projection-based visualization was generally welcomed as an alternative to common list views. However, some remarked that additional semantics of the two display axis would greatly improve orientation. Young visitors particularly liked the interactivity of the visualization whereas older ones tended to have problems with this. They stated that the reason lay in the amount of information displayed which could still be overwhelming. To address the problem, they proposed to expand only tracks in focus, increase the size of objects in focus (compared to the others) and hide the mesh overlay as the focus would be already visualized by the expanded and enlarged objects. All of these proposals have been integrated into the second prototype that is very briefly described in [19].

The second prototype was tested thoroughly by three testers. During these tests, the eye movements of the users were recorded with an Tobii T60 eye-tracker. Using the adaptive SpringLens focus, the mouse generally followed the gaze that scans the border of the focus in order to decide on the direction to explore further. This resulted in a much smoother eye trajectory than the one observed during usage of panning and zooming where the gaze frequently switched between the overview window and the objects of interest – as not to loose orientation. This indicates that the proposed approach is less tiring for the eyes. However, the testers criticized the controls used to change the focus – especially having to hold the right mouse button all the time. This lead to the introduction of the focus lock mode and several minor interface improvements not explicitly covered here.

The third prototype which is described in this paper will be tested in a larger eye-tracking study that aims to proof that the interface indeed helps during exploration.

6. CONCLUSIONS

This paper described an interactive user-interface that addresses a common problem of visualization approaches that

[5] The CeBIT is a German trade fair specialized on IT – c.f. http://www.cebit.de

are based on neighborhood-preserving projections: Mapping music collections from high-dimensional feature space onto two dimensions, projection errors become inevitable. Therefore, some objects will appear closer than they actually are and on the other side, some objects that are distant in the projection may in fact be neighbors in the feature space. The described user-interface for exploring music collections exploits the distorted neighborhood relations during user-interaction: A multi-focus fish-eye lens is used to zoom into regions of interest. While the user can control the primary focus, the secondary focus is automatically adapted. It consists of multiple smaller fish-eye lenses that focus on regions that contain tracks that are similar to those in primary focus but have been projected elsewhere. This way, the projection errors can to some extend be compensated. Further, by choosing weights for different facets of similarity, the user can manipulate the projection and the neighborhood relations visualized by the lens.

7. ACKNOWLEDGMENT

This work was supported in part by the German National Merit Foundation, the German Research Foundation (DFG) under the project AUCOMA, and the European Commission under FP7-ICT-2007-C FET-Open, contract no. BISON-211898. The authors would further like to thank all testers that provided valuable feedback for further development, Tobias Germer for sharing his ideas and code of the original SpringLens approach [17], Sebastian Loose who has put a lot of work into the development of the filter and zoom components, the developers of CoMIRVA [5] and JAudio [6] for providing their feature extractor code and George Tzanetakis for answering lots of questions concerning his MIREX '07 submission [10]. The Landmark MDS algorithm has been implemented using the MDSJ library [20].

8. REFERENCES

[1] T. Kohonen, "Self-organized formation of topologically correct feature maps," in *Neurocomputing: foundations of research*, pp. 509–521, Cambridge, MA, USA: MIT Press, 1988.

[2] I. T. Jolliffe, *Principal Component Analysis*. Springer Verlag, 2002.

[3] J. Kruskal and M. Wish, *Multidimensional Scaling*. Sage, 1986.

[4] S. Stober and A. Nürnberger, "A multi-focus zoomable interface for multi-facet exploration of music collections," in *Proc. of 7th Int. Symposium on Computer Music Modeling and Retrieval (CMMR)*, 2010.

[5] M. Schedl, "The CoMIRVA Toolkit for Visualizing Music-Related Data," tech. rep., Johannes Kepler University Linz, 2006.

[6] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle, "jAudio: A feature extraction library," in *Proc. of 6th Int. Conf. on Music Information Retrieval (ISMIR'05)*, 2005.

[7] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky?," *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, 2004.

[8] M. Mandel and D. Ellis, "Song-level features and support vector machines for music classification.," in *Proc. of 6th Int. Conf. on Music Information Retrieval (ISMIR'05)*, 2005.

[9] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," in *Proc. of 10th ACM Int. Conf. on Multimedia (ACM MULTIMEDIA'02)*, 2002.

[10] G. Tzanetakis, "Marsyas submission to MIREX 2007.," in *Proc. of 8th Int. Conf. on Music Information Retrieval (ISMIR'07)*, 2007.

[11] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[12] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York Inc, 2008.

[13] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. of 13th annual ACM Symposium on Theory of Computing (STOC'98)*, 1998.

[14] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. 12th Int. Conf. on Computer Vision (ICCV'09)*, 2009.

[15] V. de Silva and J. Tenenbaum, "Sparse multidimensional scaling using landmark points," tech. rep., Stanford University, 2004.

[16] V. de Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems 15 (NIPS))*, 2002.

[17] T. Germer, T. Götzelmann, M. Spindler, and T. Strothotte, "Springlens: Distributed nonlinear magnifications," in *Eurographics 2006 - Short Papers*, 2006.

[18] J. Nielsen, "Usability engineering," in *The Computer Science and Engineering Handbook* (A. B. Tucker, ed.), pp. 1440–1460, CRC Press, 1997.

[19] S. Stober and A. Nürnberger, "Visualisierung von großen Musiksammlungen unter Berücksichtigung projektionsbedingter Verzerrungen," in *36. Jahrestagung für Akustik (DAGA'10)*, 2010. (in German).

[20] Algorithmics Group, "MDSJ: Java library for multidimensional scaling (version 0.2)." University of Konstanz, 2009. Available at http://www.inf.uni-konstanz.de/algo/software/mdsj/.

Figure 2. Available filter modes: collapse all (top left), focus (top right), sparse (bottom left), expand all (bottom right). The SpringLens mesh overlay is visible.

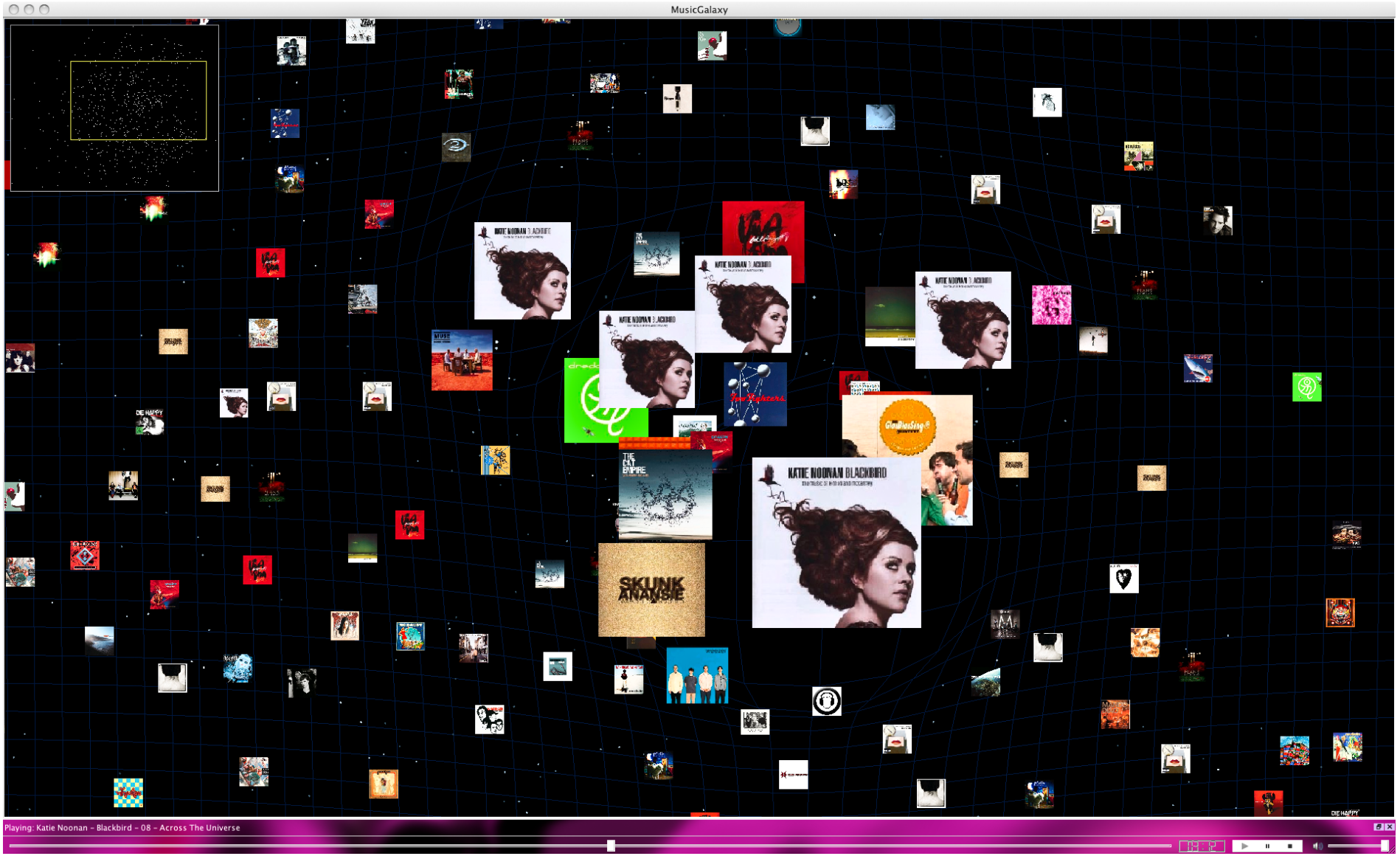


Figure 3. Screenshot of MusicGalaxy prototype with visible overview window (top left), player (bottom) and SpringLens mesh overlay (blue). In this example, a strong album effect can be observed as for the track in primary focus, four tracks of the same album are nearest neighbors in secondary focus.

FIRST STEPS IN RELAXED REAL-TIME TYPO-MORPHOLOGICAL AUDIO ANALYSIS/SYNTHESIS

Norbert Schnell
IRCAM, CNRS - STMS
Norbert.Schnell@ircam.fr

Marco Antonio Suárez Cifuentes
IRCAM
Marco.Suarez@ircam.fr

Jean-Philippe Lambert
IRCAM
Jean-Philippe.Lambert@ircam.fr

ABSTRACT

This paper describes a real-time audio analysis/resynthesis system that we developed for a music piece for ensemble and electronics. The system combines real-time audio analysis and concatenative synthesis based on the segmentation of sound streams into constituting segments and the description of segments by an efficient set of descriptors adapted to the given musical context. The system has been implemented in Max/MSP using the *FTM & Co* and *MuBu* libraries and successfully employed in the production and performance of the piece. As more and more research in the domain of music information retrieval, we use the term of *typo-morhpology* to designate the description of sounds by morphologic criteria including the temporal evolution of sound features that also can provide pertinent means for the classification of sounds. Although, the article mainly insists on the technical aspects of the work, it occasionally contextualizes the different technical choices regarding particular musical aspects.

1. INTRODUCTION

The developments described in this article have been conducted in the framework of the production of the piece « Caméléon Kaléidoscope » for an ensemble of 15 instruments and electronics by Marco Antonio Suárez Cifuentes. The piece, commissioned by the *Ensemble Orchestral Contemporain*, IRCAM, and the GRAME, has been premiered at the *Biennale Musiques en Scène* in Lyon in march 2010.

The basic idea of this work was to create a real-time system that re-orchestrates and responds to solo instruments and instrument groups using sound materials that are either pre-recorded or recorded on the fly from the same ensemble. The musical writing of the piece features densely articulated musical structures and makes intensively use of contemporary playing techniques.

The real-time analysis sub-system that we developed in this context segments audio streams into elementary events and generates a description for each segment that represents its perceived duration, its energetic evolution, and its pitch content. The same technique and descriptors are used in the real-time analysis of the playing ensemble as for the description and retrieval of pre-recorded and pre-analysed materials [1] . This choice has been made to easily allow for an extension of the system that would extend in real-time the sound data base used by the resynthesis.

We have chosen the description to efficiently represent the most essential features of the given musical material. Although it cannot be considered as a symbolic representation it permits the composer to manipulate and transform the analyzed musical phrases as well as to generate new musical phrases from the recorded material using sequence patterns and generative algorithms.

In the setup of the piece, the analysis/synthesis system described in this article is embedded into an environment of further real-time audio processing applied to the 15 instruments of the ensemble such as spatialization, transposition, and distortion. The system analyzes a single input stream that can be switched to different instruments and instrument groups. The synthesis sub-system of the system concatenates automatically generated musical phrases from the pre-recorded data base in response to the analysis of the input stream. Up to 16 synthesis voices are used in parallel.

1.1 Typo-Morphology

The principals of this analysis picks up on current trends in music information retrieval research seeking to describe sounds by the temporal development of their features [1, 2, 3, 4, 5]. This research often refers to Pierre Schaeffer's *typo-morphology* [6] that provides a framework for the description of sounds independent of their origin. In the context of automatic description of sounds of all origins, and especially in the context of music, Schaeffer's work is an excellent source of inspiration in the elaboration of abstract sound representations that capture the essential characteristics of sounds from a particular "point of view". Many authors have proposed extensions of Schaeffer's description system [7] and adapted it to a particular context of application [8].

1.2 Relaxed Real-Time Analysis/Resynthesis

The system that we developed has all properties of a real-time audio analysis/resynthesis system. It analyzes an audio input stream and can generate an output stream as a real-time transformation of the input stream. Although,

[1] The pre-recorded materials used in the piece are solo and group recordings of instrument parts extracted from an early version of the score performed by the players of the ensemble that also performed the premiere of the piece.

the segment based approach of our system makes that the information about the incoming sound stream is available only at the end of each segment, which introduces a considerable input/output latency for the system's responds.

The approach to real-time audio processing we promote with this work inscribes itself into a current trend of real-time audio processing and interactive systems in music that relaxes the latency constraints to the benefit of introducing novel sound representations into this context. The challenge of this work is to provide musically pertinent descriptions of sounds that can be calculated in real-time and to provide the means to manipulate these representations as a part of a musical work.

In this sense, we'd like to refer to this approach as *relaxed real-time* audio analysis/synthesis [2] .

This work can also be seen as an extension of existing concatenative synthesis [9] and audio musaicing [10, 11, 12] systems introducing an efficient set of descriptors representing the evolution of sound features within a sound segment.

The implementation of the system is entirely based on Max/MSP using the *FTM & Co* [13] libraries *Gabor* [14] and *MnM* [15] as well as a set of modules recently developed in the framework of the *MuBu* project [16].

2. SEGMENTATION AND DESCRIPTION

Sound descriptions create a particular "point of view" on sound materials that has to be optimized in order to fit a particular application. Similar to other domains of application, the work in the context of music composition reveals particular aspects of sound descriptions in terms of their correspondence to actually perceived sound features and, beyond that, their correspondence to a particular vocabulary of musical writing. While, for the retrieval of sounds using similarity – directly or after a transformation/manipoulation of the sound description – this correspondence can stay implicit, the more explicitly the description reflects musically relevant characteristics corresponding to the vocabulary of musical writing, the more it can be integrated into a compositional process in articulation with a written musical score.

An important challenge of this work was to optimize the representation of the audio streams as a sequence of described segments according to multiple aspects:

- Pertinence regarding the musical material (i.e. instrument sounds, style of musical writing and performance)
- Potential in terms of its manipulation as part of the compositional process
- Efficient real-time calculation of the analysis and availability of optimized analysis operators in Max/MSP

[2] Technically, the overall Max/MSP system stays a synchronous real-time audio processing system with an audio input/output latency below 50 milliseconds that also applies further (low latency) real-time transformations to the incoming sounds. Nevertheless, this approach to real-time processing also generates new requirements for the processing environment concerning the possibility to integrate different streams of processing on different levels of scheduling and possibly having different representations of time.

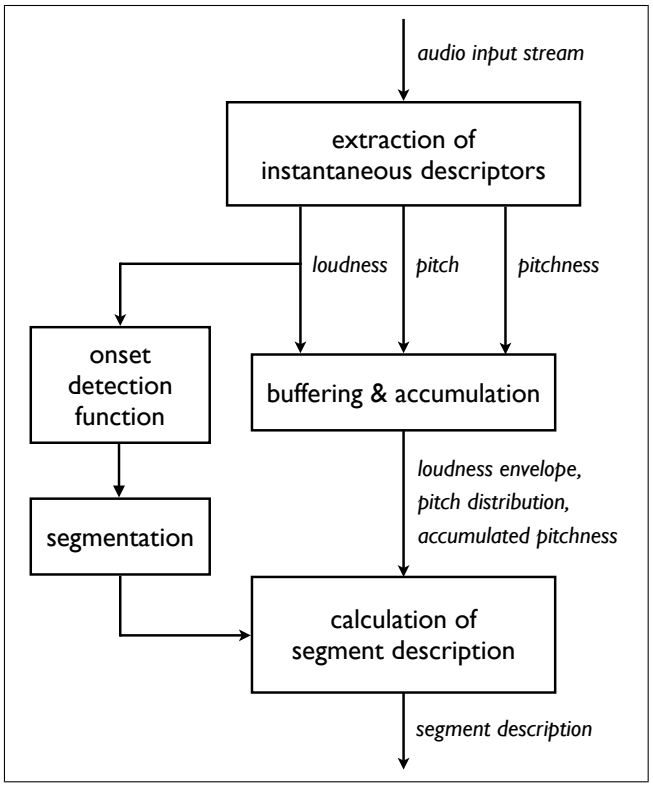


Figure 1. Schematic overview of the analysis sub-system

Figure 1 shows a schematic overview of the analysis sub-system and its different stages. As in similar systems, the analysis is organized in multiple stages:

- Extraction of instantaneous (low-level) descriptors
- Segmentation
- Calculation of the segment description

The following subsection will describe and discuss the techniques that have been employed.

2.1 Instantaneous Audio Descriptor Extraction

As in many other music information retrieval systems, the first stage of our analysis sub-system extracts instantaneous audio descriptors from the incoming audio stream. The following low-level descriptors are calculated on frames of 2048 samples with a hop size of 256 samples on the 44.1 kHz input audio stream generating regular streams of audio descriptors with a period of 5.8 milliseconds:

- Loudness extracted from the power spectrum
- Monophonic pitch
- Pitchness

The loudness is is extracted from the power spectrum using dBA weighting and the pitch and pitchness are calculated using the *Yin* algorithm [17].

The following stages of the sub-system perform segmentation and integrate the loudness and pitch information into a compact description of the segments.

2.2 Onset Detection and Segmentation

The automatic segmentation implemented in the system in based on an onset detection function calculated as the dif-

ference of loudness between the current frame and the median of the loudness over multiple previous frames (typically 3 to 5). A positive threshold is applied to this function to determine the instant of segmentation. The parameters of the algorithm are the length of the median filter and the segmentation threshold.

A variant of this algorithm had been developed on singing voice in a former unpublished research project [18]. The original algorithm had been inspired and successfully validated against other published methods [19, 20] for singing and spoken voice. Although we tested variants of the segmentation algorithm using a spectral description (i.e. MEL band and MFCC coefficients), the performance on the given sound materials was not significantly improved compared to the loudness based approach so that it would have justified the extraction of further spectral descriptions in the final system [3] .

On the given sound materials, the segmentation performs satisfactory although it evidently fails in two cases that clearly would be represented as distinct musical events in a score: extremely soft onsets (i.e. sound segments very successively appearing from silence) and smooth transitions between distinguishable events. While the latter is not an issue for the musical approach for which the system has been designed, the former remains an unsolved challenge to be further investigated on.

It was was surprisingly easy to find a parameterization of the onset detection algorithm for all used sound materials that well distinguishes local singularities of the input signal due to sound texture (i.e. roughness) from the onsets of musical events.

Theoretically, the onset detection function could also be used to determine the end of a segment by applying a negative threshold. Although, this technique has the tendency to cut off resonances and reverberations beyond the main body of particular sound events (e.g. partially attenuated chords or plates). In addition, it does not give an appropriate estimation of the actual perceived duration of the sound event. Since an overlap add technique is used for the synthesis allowing for the synthesis of almost arbitrarily overlapping segments it was not necessary to determine the end of a segment before the beginning of a the next giving the possibility to preserve resonances and reverberations wherever possible.

Consequently, a segment is defined by two successive onsets. Apart from its total duration, a set of eight descriptors representing its loudness envelope and pitch content is associated to each segment.

2.3 Loudness Envelope Description

The most essential descriptors of a segment in the given context concern the perceived energy. During the analysis, the characteristics of evolution of loudness between two onsets is recorded into a vector in order to extract the following descriptors:

- Maximum loudness
- Effective duration
- Envelope skewness

The maximum loudness represents well the perceived energy of the segment. The module used to calculate the other two descriptors actually calculates the first three standardized moments of the loudness envelope. The effective duration is derived from the spread (actually from the standard deviation) of the envelope by multiplying it with a constant factor and clipping it to the segment duration as defined by the inter-onset time. The multiplication factor that has been found by empirical experimentation (see 4) so that the descriptor represents very well the perceived duration of a segment in comparison to segments with equal or similar descriptor values and that it can be used to concatenate a sequence of sound events eliminating or equalizing the gaps between the end of one event and the beginning of the next.

The envelope skewness turns out to be an efficient and robust descriptor to distinguish whether and to which amount the perceived energy of sound event represented by a given segment raises or falls. The examples in section 4 illustrate well this descriptor.

For convenience in the processing of the descriptors, loudness is represented in the implementation by positive numbers corresponding to the actually measured loudness in dBA with an offset of 72 and clipped between 0 and 72 reducing the dynamics range used in all calculations to 72 dB.

2.4 Pitch Content Description

A second set of descriptors that describe a segment concerns the pitch. Several options have been considered for the extraction of the pitch content of a segment and the evolution of pitch within a segment. Given the sound material, the extraction of a pitch contour has been excluded. The majority of segments correspond to contemporary playing techniques without a clear pitch or multiple pitches (i.e. multiphonics or piano chords). Relatively few segments that actually have a monophonic pitch contain glissandi that would be worth to be described by a contour. The description we found describing best the pitch content of a segment is a distribution table accumulating the output of a pitch tracker over a segment.

As mentioned above, in the current version of the system we use a monophonic fundamental frequency estimation module based on the *Yin* algorithm, that also outputs a harmonicity coefficient and the sound energy (calculated as the first coefficient of the auto-correlation). For strictly monophonic harmonic sounds the module outputs precise estimation of the pitch and a quality estimation is close to 1. For slightly inharmonic or noisy but pitched sounds (e.g. due to damped resonances) the harmonicity coefficient decreases and for unpitched sounds it is close to 0. If a sound contains multiple pitches, the module tends to jump from one pitch to another still representing rather well the perceived pitches and strong resonances present in the segment. The product of the harmonicity coefficient and the energy of a frame corresponds to the relative salience of

[3] Since in our research project following up on the work described in this article we seek to include the description of timbre that anyway requires the extraction of further spectral representations, we are currently reconsidering this question.

the measured pitch. These salience values are accumulated in the pitch distribution table indexed by the estimated pitch quantized to a quarter-note scale. At an detected onset the table is evaluated for the last segment and cleared to accumulated the pitch distribution of the next segment. The pitch with the highest weight in this distribution and its quarter-tone pitch class as well as the centroid and the standard deviation calculated from the table are representing the pitch content among the set of descriptors of a segment.

In addition, the system calculates the mean value of the harmonicity coefficients for all frames with a loudness above a certain threshold (typically -72 dB) over the segment.

In summary, the following five descriptors have been chosen to represent the pitch content of a segment:

- Most salient pitch in quarter tones represented in floating-point MIDI pitches
- Pitch class of the most salient pitch
- Centroid of the pitch distribution table
- Standard deviation of the pitch distribution table
- Pitchness

These five descriptors can be seen as a compromise between descriptors that actually mean something for the user of the system (i.e. the composer) and descriptors that represent implicitly the most salient features of a segment.

3. SOUND RETRIEVAL AND RESYNTHESIS

The synthesis sub-system relies essentially on a recently developed set of modules that are designed together with an optimized data container for Max/MSP called *MuBu* [16]. In the Max/MSP implementation of the system, the data base of pre-recorded sound materials is stored in the *MuBu* data container module. The data is aligned to the audio data and associated to the onset time of the respective segment.

The interaction with the data base of pre-recorded sounds relies on a k-nearest neighbor method. The module used in the system implements a KD-tree allowing for an efficient retrieval of the segments of the data base of pre-recorded sounds that comes closed to a given description. For the retrieval, each descriptor can be given a weight to define its importance or even to exclude it from the query. The description for querying sounds can be directly given by the analysis of an audio input, generated arbitrarily or by transformation of the output of the analysis.

A basic analysis/resynthesis system is created when using the descriptors generated by the analysis of an input stream in real-time directly for the query of a the sound with the closest description in the data base and playing immediately the retrieved sound segment. The result of this setup is an output sound concatenated from sounds present in the data base that reflects at the same time the material in the data base as well as the "point of view" on the sound material – and sound in general – that is incarnated by the set of descriptors [4] . In the design phase of the system, we have intensively made use of this simple setup to permanently evaluate the performance of the system regaring the pertinence of the chosen description.

The most basic transformation of descriptors that we are experimenting with is the scaling of descriptors values. Scaling allows, for example, for adapting the range of the descriptors produced by the analysis of one instrument or playing style to another. We have calculated for each instrument in the data base the mean and standard deviation as well as the minimum and maximum values of each descriptor over all segments. Dependent on the descriptor and the desired effect one would use either the extreme values or the statistical distribution as a reference for scaling.

4. EVALUATION AND DEMONSTRATION

The system has been permanently evaluated in listening experiments during its design. In addition to these experiments that compared the behavior of the system for different sound materials, we have developed a small environment around the analysis sub-system to visualize the analysis data in real-time and offline in Max/MSP. This application records the loudness envelope and the pitch distribution tables as well as the onset times of the segments in addition to the descriptor values. The recorded data and the waveform of the corresponding sound segment can be visualized from multiple points of view using a dedicated graphical module of FTM & Co and played back with different options [5] .

The fact that programming, real-time and offline analysis/resynthesis, and visualization are possible in the same environment significantly facilitated the design and implementation of the system. Although this is common for many applications using Max/MSP (or similar environments) as rapid prototyping and execution environment, the availability of a large number of efficient analysis and statistical modeling operators and visualization tools in the FTM & Co libraries adds to this an additional dimension for the design of analysis/resynthesis systems.

The screen-shots presented in the figures 2 to 6 show segments analyzed by the system that are representative for the sound materials that we have worked with and that permits to briefly discuss the representation of the sound segments by the chosen set of descriptors. We selected different instruments and playing modes.

Each figure shows the loudness curve and pitch distribution table superposed to the waveform of the corresponding segment as well as the nine descriptor values calculated for the segment. While the standard deviation of the pitch distribution table has been calculated on the original values in a scale proportional to the power of the signal, in the figures below the values are visualized in logarithmic scale. The effective duration is additionally marked with a small vertical line crossing the zero-level of the waveform display.

Even if the segments may appear isolated they are in fact all real-time segmentations of the original recordings.

[4] Since the vector of descriptors is output by the analysis stage at the end of each segment the timing (i.e. rhythm) of the synthesized sound does not correspond to the input sound unless the onset times are corrected using an estimated maximal segment duration.

[5] For the purpose of this publication the visualization has been simplified and superposed in a single window.

The musical writing of the scores used for the recordings is rather dense, but privileges very clearly articulated relatively short events over longer sounds with a continuous evolution.

Figure 2 shows a pitched marimba note. The pitch distribution is reduced to a single peak at the pitch of the note (69) that also corresponds to the low standard deviation value (0.26). The envelope skewness (0.54) indicates a decaying loudness. The relatively low pitchness value corresponds to the inharmonicity of the marimba sound.

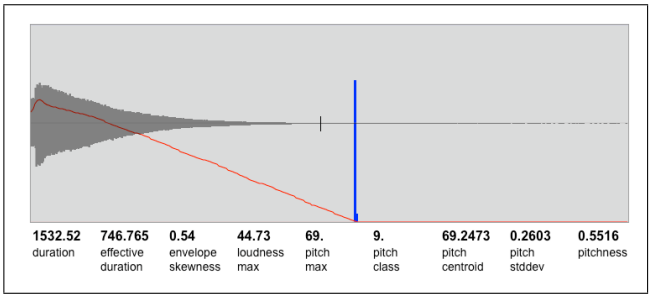


Figure 2. Waveform, and description data of a segment of a marimba note.

The pitch distribution of the segment of a flute note played with flatterzunge shown in figure 3 represents well the two salient pitches that are audible when listening to the sound. The descriptors calculated from the distribution still represent the pitch of the note (75.5) derived from the maximum of the distribution, but reduce the two pitches to an augmentation of the standard deviation (2.94) and a centroid below the maximum pitch. The low value of the loudness envelope skewness (0.11) indicates a rather flat envelope.

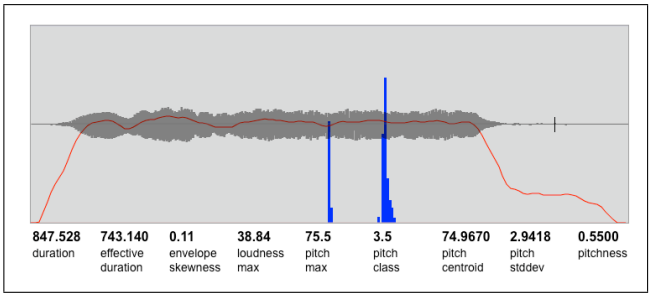


Figure 3. Waveform, and description data of a segment of a flute note with flatterzunge".

The segment represented in figure 4 corresponds to a clarinet multiphonic emerging from silence with a strong crescendo. The note onset of this example is strong enough to be detected by the system. Apart from the strongest peak that in fact corresponds to the strongest perceived pitch, the pitch distribution in this example only very vaguely corresponds to the pitch content of the multiphonic. Nevertheless, some overall characters of the pitch content are represented by the high standard deviation (5.8) combined with a high pitchness value (0.82). The crescendo is well represented by the very low negative skewness value (-1.17). The segment is cutoff by the onset of a staccato note following the crescendo.

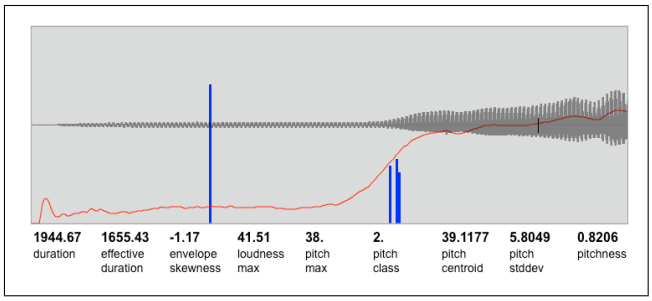


Figure 4. Waveform, and description data of a segment of a clarinet multiphonic.

The pitch distribution of the trombone glissando in figure 5 represents well the pitch range of the glissando. The additional peaks on the left and the right are octave errors of the pitch estimation that do not significantly change the standard deviation of the pitch distribution calculated on the linear scale values of the table (2.04) corresponding to 2 semitones. The slight crescendo over the segment is expressed in the negative skewness value (-0.22).

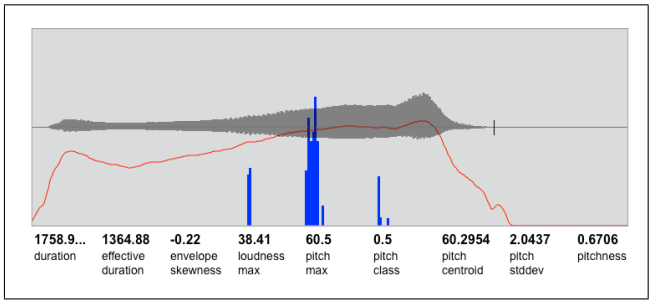


Figure 5. Waveform, and description data of a segment of a trombone glissando.

The last example in figure 6 visualizes the segmentation and analysis of a short bass note played “ecrasé”. Even though the effective duration is correctly estimated its display starting from the beginning the segment is not appropriate in this case. The low pitchness value (0.26) witnesses of the noisy character of the sound segment that corresponds to a note in the middle of a staccato sequence. Nevertheless, the most salient perceived pitch of the segment is represented by the indicted maximum of the distribution (35.5).

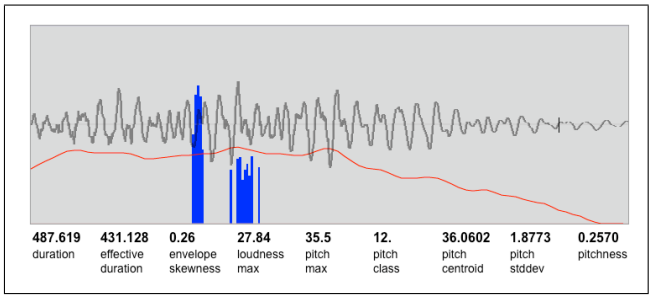


Figure 6. Waveform, and description data of a segment of double bass note played “ecrasé”.

5. CONCLUSIONS AND CURRENT RESEARCH

With the work described in this article, we have achieved a first step in the development of a real-time analysis/resynthesis using a segment based description.

Staying rather simple, the description developed for the context of a contemporary music piece has been proven its efficiency in the context of the given musical production and opened for us an interesting field of further investigation.

Although the system does not perform an explicit classification, it permits to access to the data base by typomophological criteria.

We are currently working on several improvements and extensions of the system. The most important improvements mainly concern the detection of soft note onsets and smooth note transitions as well as the development of a compact description of timbral aspects.

Further experiments and developments concern the transformation of the segment description and the resulting possibilities in composition. For example, the descriptor values can be normalized and scaled or inverted in order to create corresponding variations in the retrieval of sound segments and resynthesis.

While we currently do not apply any analysis and modeling of the sequence of segments, the given representation has an interesting potential to be used in conjunction with techniques such as *Bayesian networks* and *Factor Oracle*. Sequence modeling may require a more explicit classification of segments that can be easily derived by clustering in the descriptor space.

Even if a segment based approach was an obvious choice regarding the very strongly articulated character of the piece we are currently considering alternative techniques that do not require a segmentation at the analysis stage and still allow for the manipulation and retrieval of musically relevant sound segments represented by a temporally integrated description corresponding to the temporal evolution of sound features.

6. REFERENCES

[1] J. Ricard and H. Perfecto, "Using Morphological Description for Generic Sound Retrieval," in *International Conference on Music Information Retrieval (ISMIR)*, 2003.

[2] J. Ricard and H. Perfecto, "Morphological Sound Description Computational Model and Usability Evaluation," in *Proceedings of the AES Convention*, 2004.

[3] G. Peeters and E. Deruty, "Automatic Morphological Description of Sounds," in *Proceedings of Acoustics 08*, 2008.

[4] G. Peeters and E. Deruty, "Sound Indexing Using Morphological Description," in *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.

[5] J. Bloit, "Interaction musicale et geste sonore : modélisation temporelle de descripteurs audio," PhD Thesis, 2010.

[6] P. Schaeffer. Paris, France: Seuil, 1966.

[7] M. Chion. Paris, France: INA/GRM, 1966.

[8] L. Thoresen and A. Hedman, "Spectromorphological Analysis of Sound Objects: An Adaptation of Pierre Schaeffer's Typomorphology," *Organised Sound*, vol. 12, pp. 129–141, 2007.

[9] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2006.

[10] T. Jehan, "Creating Music by Listening," PhD Thesis, 2005.

[11] M. Casey, *Soundspotting: A New Kind of Process?* Oxford University Press, 2009.

[12] P. A. Tremblay and D. Schwarz, "Surfing the Waves: Live Audio Mosaicing of an Electric Bass Performance as a Corpus Browsing Interface," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, June 2010.

[13] N. Schnell et al., "FTM — Complex Data Structures for Max," in *Proceedings of the International Computer Music Conference (ICMC)*, Septembre 2005.

[14] N. Schnell et al., "Gabor, Multi-Representation Real-Time Analysis/Synthesis," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Septembre 2005.

[15] F. Bevilacqua, R. Muller, and N. Schnell, "MnM: A Max/MSP Mapping Toolbox," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, Mai 2005.

[16] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi, "MuBu & Friends - Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP," in *Proceedings of the International Computer Music Conference (ICMC)*, August 2009.

[17] A. de Cheveigné and H. Kawahara, "YIN, A Fundamental Frequency Estimator for Speech and Music," *JASA*, vol. 111, pp. 1917–1930, 2002.

[18] J.-P. Lambert and N. Schnell, "Internal Report of the ANR Project *VoxStruments*," tech. rep., IRCAM, Mai 2009.

[19] P. Brossier, J. P. Bello, and M. D. Plumbley, "Real-time Temporal Segmentation of Note Objects in Music Signals," in *IEEE Transactions on Speech and Audio Processing*, 2004.

[20] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, D. M., and M. B. Sandler, "A Tutorial on Onset Detection in Music Signals," in *IEEE Transactions on Speech and Audio Processing*, 2005.

INTERPRETATION AND COMPUTER ASSISTANCE IN JOHN CAGE'S *CONCERT FOR PIANO AND ORCHESTRA* (1957-58)

Benny Sluchin
Ircam
benny.sluchin@ircam.fr

Mikhail Malt
Ircam / MINT – Paris IV - Sorbonne
mikhail.malt@ircam.fr

ABSTRACT

Conceptual musical works that lead to a multitude of realizations hold a particular interest. One can't talk about a performance without taking into account the rules that lead to the existence of that particular presentation. After dealing with similar works of open forms by Iannis Xenakis and Karlheinz Stockhausen, the interest in John Cage's music is evident. His works are "so free" that one can play any part of the material; even a void set is welcomed. The freedom is maximal and still there are decisions to consider in order to perform the piece.
The present article focuses on the *Concert for Piano and Orchestra* of 1957–58 [1], and it is part of the *Cagener* project, intended to develop a set of conceptual and software tools, which generates a representation of the pieces, intended to assist the performers in their task. The computer serves as a partner in making choices of multiple possibilities, mix together sounds of different sources and of various kinds and following compositional ideas clearly stated.

1. INTRODUCTION

The performer approaching John Cage's music composed after the middle of the 20th century is often surprised to encounter a large amount of freedom mixed with a set of precise instructions. As a common result, the musician will determine "a version" in which he will decide on the free elements included in the score. A fixed score is thus created and used repeatedly. The performer will play it without any doubts of the composer's intentions. In fact, most of Cage's scores composed after the fifties are not to be pre-generated. Each performance should be unique and undetermined. Using the computer helps one to perform, ignoring what and when he is going to play.

2. COMPUTER-AIDED PERFORMANCE

The musical world offered itself a multitude of tools with the evolution of computer technologies. At first, dedicated to an employment in musical composition, they were oriented and adapted to a use in musical analysis and as aid tools to interpretation.

Several practices concerned with the interpretation field were developed. One can mention:

• The use of audio and MIDI sequencers as "super metronomes". It is common today that interpreters enter complete scores in sequencers as a way to work out difficulties in the performance (concerning especially contemporary pieces). The musician can thus progressively work the problematic passages by varying the speed; he can approach comfortably various tempi changes in combination with eventual rallentandi and accelerandi.

• The use of sequencers or notation programs to practice playing in ensemble. This is a logical extension of the "Minus-one" idea.

• The use of dedicated tools capable of correcting the player's interpretation.

An increasing number of composers prepare interpreters' oriented computer programs in order to help them perform with the computer before starting with the actual musical piece.

There are other examples of computer tools created by or for interpreters, but our concern here is to show a new field developed in the last twenty years.

In our topic here, the interpretation of a category of Cage's work, in which the concepts of liberty and indetermination are predominant, it seems that the paper aspect of the scores is an obstacle in the realization. The wish that the player could navigate freely, non-determined and without restraint, through the musical material seems not helped by the fact that the music is presented on paper, and thus in a determined order. Computers may bring a solution to that particular difficulty for Cage's and also other composers' music. The actual playing prevents the musician from doing other tasks to orient his choices in "real-time". For example Iannis Xenakis in *Linaia Agon* (trio for horn, trombone and tuba, 1972) asks for a passage where the different instrumental choices are directed by a "gain matrix". The choice is computer-aided in order to enable a smooth interpretation [2].

One aspect of the tools proposed here is that they are oriented towards interpretation. In that concern, the interface should "contain" implicitly or explicitly all the instructions, constraints and concepts defined by the composer, as they will establish an "experimentation

field". For the construction of Computer-aided performance (CAP) tools, the careful study of the pieces of John Cage and its formalization is necessary. The final interface will be, in a certain way, a computer model of the particular piece.

3. THE *CONCERT FOR PIANO AND ORCHESTRA.*

3.1. Musical Context

The *Concert for Piano and Orchestra* (1957–58) marks a decisive step towards the definition of the notion of « indeterminacy » and appears to be one of the more important works of John Cage, a milestone in his path. For the first time, control over decisions regarding all aspects of music is given to interpreters. Each execution may well sound differently from one another, and duration may vary each time. It is no "Concerto" for piano and orchestra, but a chamber ensemble piece whose instrumentation is to be defined at each performance. There are parts for thirteen instruments (three for violin, two for violas, one for cello, one for contrabass, one for flute who doubles on a piccolo and alto flute, one for clarinet, one for bassoon doubling on the baritone saxophone, one for trumpet, one for trombone and one for tuba), solo for piano and conductor. The individual parts are all "Solos" meaning that there is no relationship or coordination between them. Any portion of it (also void) may be chosen to be played. A version of *Concert* may thus have any number of these instruments including none, which will, in that case, be a silent version as in *4'33"*.

3.2. Score Description

In *Concert for Piano and Orchestra* there is no full score. There are 2 conductor's score pages, 63 pages for piano, and 184 instrumental pages score (Table 1)

Violin 1: pages 1–16
Violin 2: pages 17–32
Violin 3: pages 33–48
Alto 1: pages 49–64
Alto 2: pages 65–80
Trumpet in Bb: pages 81–92
Violoncello: 93–108
Tuba in F and Bb: pages 109–120
Clarinet in Bb: pages 121–132
Flute, Piccolo, Alto flute: pages 133–144
Bassoon, Saxophone: pages 145–156
Double Bass: pages 157–162
Sliding Trombone: pages 173–184

Table 1: Instrumental parts

3.3. Instrumental Scores

Each instrumental part (each Solo) is composed of one page of detailed instructions to performance and of a collection of pages containing musical events, called "notes" by Cage. Figure 1 shows the third page of the trombone score.

Each instrument page is a collection of punctual musical events, (see Figure 2 and Figure 3), displayed on 12 to 16 music sheets. Each event is a compound one, having relative pitch, dynamics, playing modes, and other indications.

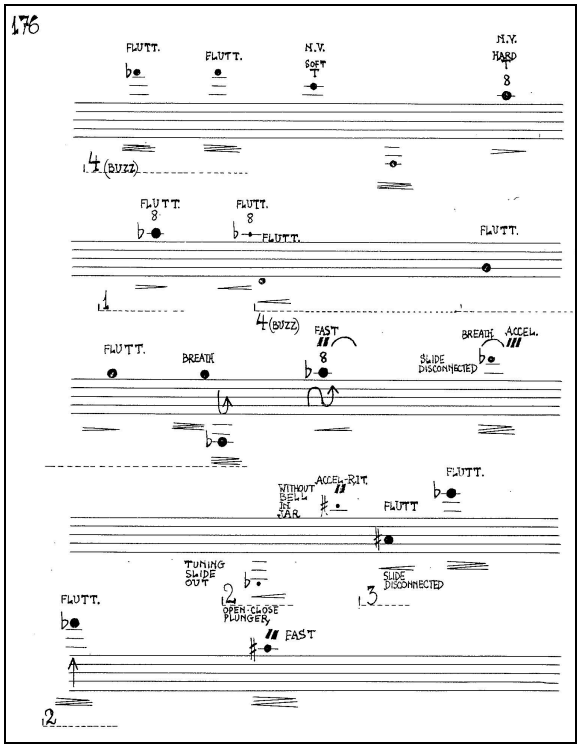


Figure 1: Page 176 of trombone score

The events are distributed variously on each sheet, from extreme density (music system with around 10 events) to empty staves (i.e. trombone score [1, p. 178]).

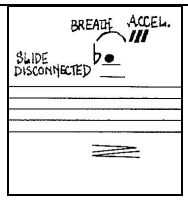


Figure 2: 5th event 3rd system of page 176, from trombone part.

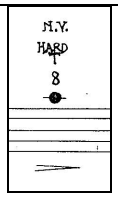


Figure 3: 5th event 1st system of page 176, from trombone part.

3.4. The Piano Score

The piano part consists of 63 pages; "each page is one system for a single pianist". The elements of a page are musical structures that were generated using different composition techniques, some varieties of same species or altogether different (Figure 4). Sometimes one structure is too long and continues on several consecutive pages, as for example, the structure "B" in page 1 [1, p.4] (Figure 4).

3.5. The Conductor's Score

Cage has also planed a conductor's score, but not in the traditional way. A conductor, providing the function of clock may participates in the performance of *Concert*. He turns his arms like a clock. The time shown by the conductor may be compressed, widened or literal physical time (His arms may not turn with the same velocity of a regular clock). Cage provides a table with what he calls "clock time" (the time shown) and the "effective time", from which the conductor prepares his conducting score.

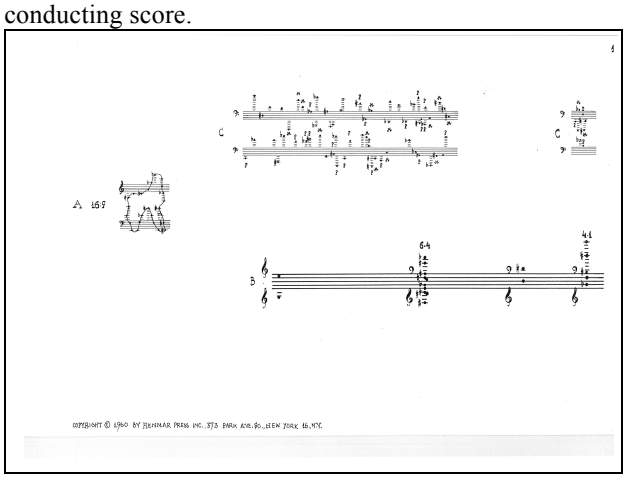


Figure 4: Page 1 from the piano part.

3.6. "Concert" model

Even if each instrument has his own "instruction page" some elements are common to all. Each player is free to play any elements of his choice, wholly or in part and in any sequence. In this manner, each presentation is unique, and Cage considers the piece as "in progress" [3].

4. PERFORMING CONTEXT

4.1. The main concepts in John Cage's early pieces

In the John Cage's *Concert* performance four concepts are highlighted: Silence, Chance, Indeterminacy and Unintentionality.

4.1.1. Silence

In connection with his encounter with Zen Buddhism [4], Cage rethinks his understanding of music. As a result, he composes *4'33"* [5], a work whose abandonment of intentional sound production drew controversy to his compositions. Cage spoke of silence in a new and positive way. Not only has it an importance in the creation of structure but one has to think of it not as an absence of sound but as a presence to fill an acoustical space.

At first, Cage developed a structural concept of silence, considering it as an absence of sound helping to structure the music by its alternation with sound. The silence between the notes gave the work its cohesion.

"Formerly, silence was the time lapse between sounds" [6, p. 73]

In the *Concert* instrumental parts instructions Cage asks:

"All notes are separated from one another in time, preceded and followed by a silence (if only a short one)."

Later, Cage adopted a spatial concept of silence, in which it was composed of all the ambient sounds that together formed a musical structure. Finally his concept evolved towards viewing silence as non-intention. Both sound and silence would exist only in the non-intention manner of nature [7]. As we will show further, this concept will appear clearly in the instructions given to the performers. It will be considered as an esthetic object as well as a concept not be forgotten in the computer solution presented.

4.1.2. Indeterminacy

The principal of indeterminacy allows the performers to work independently from each other. In this way, the musician ignoring the output of his fellow musicians will concentrate on his own part and the set of instructions, which imposes concentration even if degree of the freedom involved is high [8].

« Bringing about indeterminacy is bringing about a situation in which things would happen that are not under my control. » [9, p. 109]

Another meaning of indeterminacy is the fact that the final result is not controlled by the composer himself and the result is partially produced by a chain of performer's un-intentional choices.

4.1.3. Chance

For John Cage, "chance" is the set of random or non-determinate processes used in the composition itself. In this way he applied the concept of non-intention in the musical material choice in his own composition process.

"Variation in gongs, tom toms, etc. and particularly variation in the effects on pianos of the use of preparations, prepared me for the renunciation of intention and the use of chance operations." [11, p.91]

While *chance* is related to the compositional process, indeterminacy is related to the composer's lack of control of the performance.

"James Pritchett provides a succinct distinction between indeterminacy and chance: while chance "refers to the use of some sort of random procedure in the act of composition," indeterminacy "refers to the ability of a piece to be performed in substantially different ways."[16, p. 61]

This clearly shows the difference between these two concepts.

4.1.4. *Unintentionality*

We could find the genesis of the unintentionality concept in this Cage's quote:

"Improvisation . . . that is to say not thinking, not using chance operations, just letting the sound be, in the space, in order that a space can be differentiated from the next space which won't have that sound in it. I'm perhaps too young at this work to know how to describe it." [10, p. 582]

The unintentionality concept is the idea that a performer produces sounds, musical events without intention, aim, purpose, reason or given meaning. As Cage tell us, it comes from his Zen Buddhism studies:

"..., it was rather my study of Zen Buddhism. At first, my inclination was to make music about the ideas that I had encountered in the Orient. The String Quartet [1950] is about the Indian view of the seasons, which is creation, preservation, destruction, and quiescence; also the Indian idea of the nine permanent emotions, with tranquility at the center. But then I thought, instead of talking about it, to do it; instead of discussing it, to do it. And that would be done by making the music nonintentional, and starting from an empty mind. At first I did this by means of the Magic Square." [11, p. 94]

4.1.5. *Cage and improvisation*

« ... I have avoided improvisation through most of my work. Improvisation seemed to me necessarily to have to do with memory and taste, likes and dislikes. » [10, 581]

The improvisation for Cage seems to be related to the unintentionallity :

"Improvisation . . . that is to say not thinking, not using chance operations, just letting the sound be, in the space, in order that a space can be differentiated from the next space which won't have that sound in it. I'm perhaps too young at this work to know how to describe it." [10, 581]

4.2. Performance Problems

Analyzing the Cage instructions and his musical and aesthetical points of view, we realize that the traditional scores make the players performance difficult, especially regarding the "Unintentional" choices of different musical objects. The material and sequential aspect of paper scores is an obstacle to the realization of the Cage's main idea, that the player could go thru, freely, without constraint and without intention through the score.
As it is not always easy to jump quickly between two musical elements found on separate pages, one will tend to an interpretation privileging the grouping of objects belonging to the same page.

5. COMPUTER MODELING, MODELS DESCRIPTION

5.1. From concepts to reality

How could one help the player, as well as possible, to perform the score in a context of "indetermination" and "not-intentionality"? In what manner could one enable him to represent the Cage's musical thought?
It is the freedom relationship pre-determination that gives the player the main problem. Even if we find very hard instrumental passages, the main difficulties are: making the choice of when and what is to be played, what order to choose for the elements, the amount of silence to insert between the events, and all this while ignoring the output of the other musicians involved. It has to be kept in mind that by the absence of intention, one should also ignore what he himself is about to perform. This means, that the entire score should be at the player's disposal, and that he will make up his mind intuitively and spontaneously. One possible solution was to provide an adapted interface. Here the choice is not only of timing but concerns the material itself.

5.2. The three interfaces

Three kinds of interface, ways displaying music, were proposed.

5.2.1. *Interface A*

Here (Figure 5, Figure 6), the pages are displayed as Cage conceived them originally. The interpreter may turn pages sequentially or skip randomly between them. The music to his disposal may contain the entire score or a subset of it. The choice of what to play is made during performance. The interpreter may play or not these musical events, and it is up to him to manage the time allocated to each page or sequence of elements.

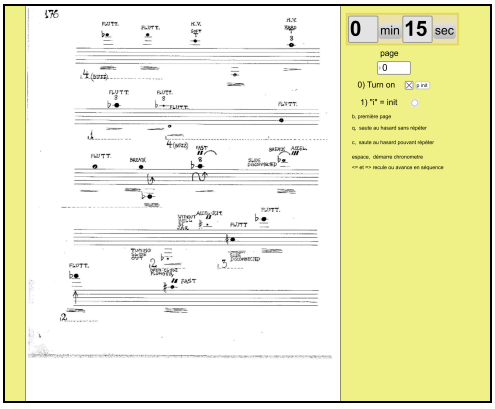


Figure 5: Interface A (trombone score)

The main advantage is the fact that the performer can navigate easily through the score. The graphic disposition of the various events in the paper score space may influence the performer. Does there exist a

cognitive weight related to their position in the original page, which would influence the choices? Do there exist "hot" and "cold " zones "in the graphic space of the partition? Some questions remain.

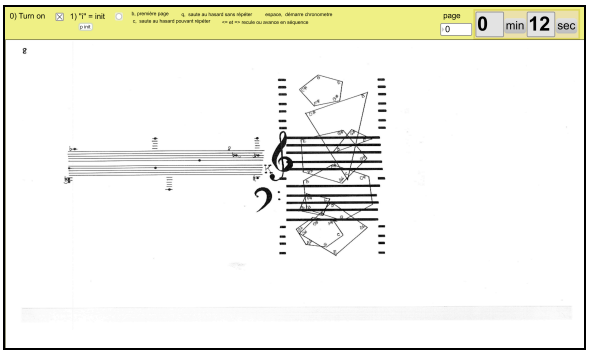


Figure 6: Interface A (piano score)

5.2.2. Interface B

Here (Figure 7), one event at a the time is displayed. The interface proposes two modes. In the first one, the event appears "at time" it is to be played. The performer is in a position of concentration, waiting for the event display, without knowing which one will appear. The event remains displayed during a short time (which can be parameterize). In the second mode, the player is notified of the next following event. He ignores though when this one will appear. In this way it helps him to be prepared (instrument change, remove or insert a mute, tune or detune the instrument, etc).
This interface leaves little control to the player. However, it has the advantage to embody the "Indetermination" and the "Unintentionality" asked by Cage, and to free the interpreter from the "physical" limits of paper score.

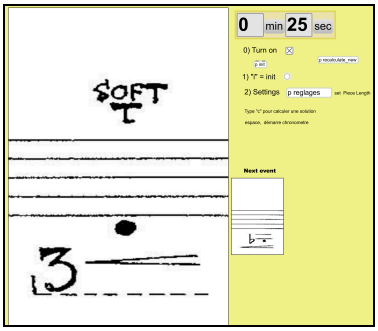


Figure 7: Interface B

5.2.3. Interface C

Here (Figure 8), we look for displaying the events in time. With this interface, it is possible to build scores generated algorithmically, the "elements" of the score being reorganized according to various calculation modes. Calculation procedures are based either on the "instructions" given by Cage to the performers or on other methods enabling to reorganize and give "directions" to the musical material.

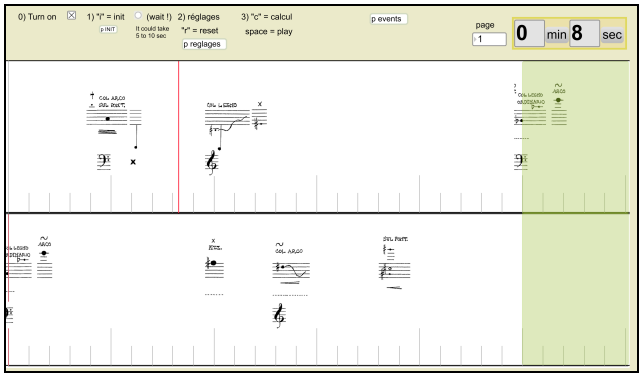


Figure 8: Interface C (violoncello score)

The setting window (Figure 9) allows changing the time displayed by page and some parameters for the score generation like the total length of the piece. Cage indicates: "given a total performance time-length, the player may make any program (including additional silences or not) that will fill it".

In addition to the idea of freeing the player from the physical constraints of the paper score, and of taking in account the "Indetermination" and "Unintentionality", this interface also makes it possible for the player to prepare himself more efficiently for the various changes of instrument configuration and the various playing modes. Besides this, it enables the interpreter to use his own will at the moment of the execution, by choosing to play or not the proposals presented.

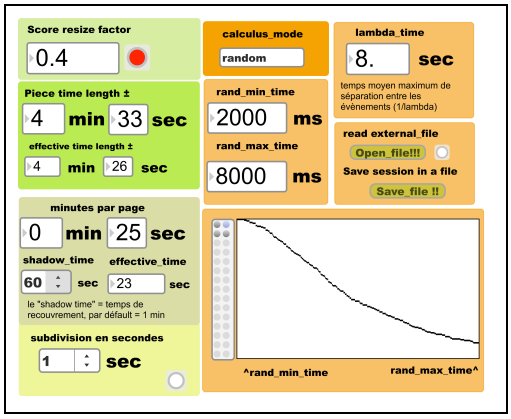


Figure 9: Setting window for the interface control

5.3. The conductor's interface

The conductor's interface is perhaps the easiest interface to build. Actually it consists of a clock interface (Figure 10). The data coming from this interface will be sent to the player's computers to drive their chronometers.

5.4. Score Calculus

To construct a computer version, two main dimensions were calculated: the punctual events order and an events distribution in time.

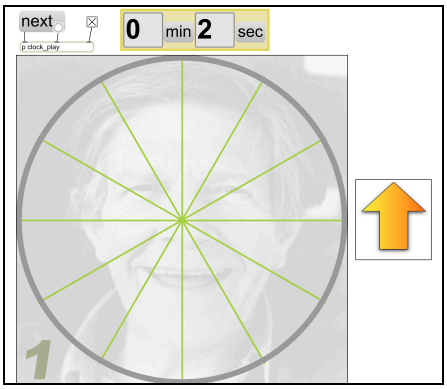


Figure 10: Conductor's interface

5.4.1. *Cournot model of chance*

The first mean of calculus was based on a "unintentional choice" modeling, based on a "Cournot concept of chance"[1], as an aid to the performer to avoid intentional or conscious choices. From this point of view a sequence will be a combination, a juxtaposition (a coincidence) of two independent and deterministic values sequences. In our specific case, it means that a computer calculated version would consist of a sequence of events and a sequence of time positions. Each one of these sequences will be calculated independently.

For the events organization we used a "uniform random choice" with or without event repetition allowed, and for the time structure three methods:

1) A "uniform random choice" of time intervals between the events (Times between events distributed uniformly in the interval {*Rand_min_time*, *rand_max_time*})

2) An "exponential random choice" of time intervals between the events (Times between events distributed according to exponential distribution of parameter "λ ")

3) A "random choice" of time intervals between the events (According to a probability distribution function built by the user with boundaries {*Rand_min_time*, *rand_max_time*})

To determine the temporal positions of each cell we based our calculations on time "between" the cells rather that directly on the positions in time. This decision enabled us to model more efficiently and to take in account one of the main instructions of Cage concerning the need for "silences" between the events (4.1.1).

5.4.1.1 *The "time" models*

As a time model, we explored three ways. The first one was a single uniform stochastic distribution scaled between a minimum and a maximum time interval {*Rand_min_time*, *rand_max_time*}.

The second one, inherited from Iannis Xenakis's experiments at the end of the 1950s [13, p. 26, 169, 171, 243-246], where a time model based on the exponential distribution to calculate the probability of a musical event having a given time length.

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & ; \; x \geq 0 \\ 0 & ; \; x < 0 \end{cases} \qquad (1)$$

Where:

λ: is the average density of events by length unity.

To implement this we did a javascript *Max/MSP* object using the expression:

$$\delta(\lambda) = \frac{\ln(\sigma).-1}{\lambda} \qquad (2)$$

with σ being the result of a uniform random variable between 0 and 1.

The third way was using self-made distributions using the <itable> *Max/MSP* object (Figure 11), where different curve shapes are investigated.

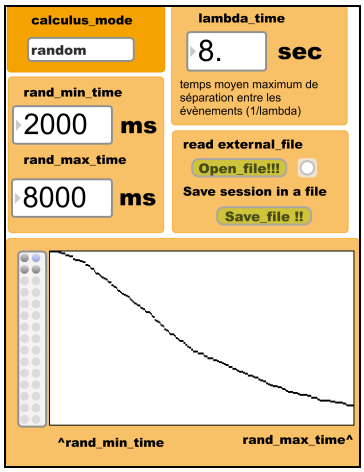


Figure 11: interface for building our own probability distribution

5.4.2. *Deviations and Variations*

It was clear to us that modeling the *Concert*, is a pragmatic way to study and to try to understand the Cage's musical mind. It is why we propose to analyze score calculus ways that are the complete opposite of the first one. Instead of generating an unpredictable score, we try to determine one, using (if possible) all the implicit information given by Cage. Our main purpose is to try to answer the question "how far could we get away from Cage's instructions without breaking the original model". How "intentionality" could change the score? How it could change the Aesthetic.

As for the first way of calculus, two main dimensions were calculated: the punctual events order and an events distribution in time.

[1] « Les évènements amenés par la combinaison ou la rencontre de phénomènes qui appartiennent à des séries indépendantes, dans l'ordre de la causalité, sont ce qu'on nomme des évènements fortuits ou des résultats du hasard ». [12, p. 73 ¶40]

5.4.2.1 Ordering events according with a description vector space

The instrumental parts of *Concert* give more elements to model. Looking closely at each event we have attached it to a set of characteristics. For example in the trombone part (also known as *Solo for Sliding trombone*) we identified:

1. Placement (position in the score staff)
2. Nature of event (played normally, tuning slide out, mouthpiece in bell, spit valve open, without bell, without bell in jar, with slide disconnected, conch, mouthpiece with mute, and conch with mute)
3. Pitch, represented as a MIDI pitch.
4. Head Size of notes (small, medium, or large)
5. Dynamic profile (nothing, crescendo, diminuendo, both)
6. Articulation (nothing, breath, soft Tongue, hard Tongue)
7. Vibrato (with or without)
8. Formant (coloration of the sound when sustained: nothing, fluttertongue, double and triple tongue, trills etc.)
9. Formant speed (rit., accel., rit.-accel., accel.-rit., fast, slow)
10. Mute (without, straight, plunger, cup, buzz, hat, plunger open close)
11. Arrows & curves (smaller microtonal slides, no arrow, curve down, curve up, arrow down, arrow up, etc.)

From this information we built a "descriptor vector space" where each component vector had the follow structure:

$$V_{BD\_i} = \left(s_i, p_{BD\_i0}, p_{BD\_i1}, p_{BD\_i2}, ..., p_{BD\_i10}\right) \quad (3)$$

where

s_i is a symbol identifying a particular event

$p_{BD\_i\ j}$ is the value of the "*jth*" descriptor for the "*ith*" vector.

At the same time we built (in the *OpenMusic* composition assisted computer environment) a sequence of "target vectors" in the form $V_c[t]$, with t representing time, where each vector has the follow structure:

$$V_c[t] = \left(s[t], p_0[t], p_1[t], p_2[t], ..., p_{10}[t]\right) \quad (4)$$

with each parameter descriptor having a dynamic evolution in the time.

For each discrete time value $[t]$ we calculate the vector

$$V[t] = \min\left\{dist\left(V_c[t], V_{BD\_i}, \omega[t]\right)\right\} \quad (5)$$

where

$$\omega[t] = \left(\omega_0[t], \omega_1[t], \omega_2[t], ..., \omega_{10}[t]\right) \quad (6)$$

it is a weight vector. As distance functions we used weighted Euclidian and Chebythchev distances.

$$dist_{euclid}(p,q,\omega) = \sqrt{\sum_{i=0}^{10}\left(p_i - q_i\right)^2 * \omega_i} \quad (7)$$

$$dist_{euclid}(p,q,\omega) = \max_i\left(\left|p_i - q_i\right| * \omega_i\right) \quad (8)$$

Obtaining, in this way, a sequence $V = \left\{V[0], V[1], V[2], ..., V[n]\right\}$ of $V[t]$ vectors (equation 4). From this sequence a symbol sequence $S = \left\{s_0, s_1, s_1, ..., s_n\right\}$ is obtained. Each s_k being the first dimension of the correspondent $V[t]$ vector (equation 5).

Chebythchev distance showed results where the vector derived from equation 5 contains almost one parameter p_i that corresponds to one of the parameters of target vector (equation 4). This will lead us to sequences that map better with the target vectors evolution. With Euclidian distance, as the minimum distance returned is a sort of mean distance from the target vector, without any need to contain explicitly a p_i parameter, the vectors obtained, could be very far (from a musical point of view), from what is asked in the target vectors evolution.

5.4.2.2 Time evolution

As for the first way of calculus we used basically the exponential distribution (see equation 1) but with a lambda parameter as a time function, $\lambda = \lambda[t]$.

5.4.2.3 Exporting data

These calculi were made in the *OpenMusic* environment and exported to be read in our Max computer interface.

In that way, one can generate punctual event organizations according to one or more constraints on the different characteristics. One may give it as input curves or functions that describe temporal evolution characteristics, or as textual constraints represented as logical expressions. This part of the work represent a lot of interest, as the player is unable to deal with such tasks during a performance. The computer output can still be regarded as a proposition of a "version" from which the performer still has his choice. This logical part of the project is actually implemented in *OpenMusic* environment, but it will be ported very soon in the *Max/MSP* environment.

6. PERFORMANCES

Two performances where given. "Triton" (Les Lilas, France, May 2th 2009), with Fabian Fiorini (piano), Guillaume Orti (alto sax), Benny Sluchin (trombone),

Eric-Maria Couturier (violoncello) and Mikhail Malt (computer and video improvisation).
"Hateiva » (Jaffa, Israël, October 29th 2009) Amit Dolberg (piano), Yonatan Hadas (clarinet), Benny Sluchin (trombone) and Eric-Maria Couturier (violoncello).

7. CONCLUSIONS AND PERSPECTIVES

The construction of computer models of musical pieces is not a neutral process. It is fundamental to know well the works under study, understand the constraints left by the composer, as well as the historical context of its creation. But these are still insufficient in the modeling process. Every musical work has a part of liberty and ambiguity. These "holes" must to be filled up to enable the modeling process. One has to take decisions as a function of a work assumption, founded on musical and musicological bases. The necessity to represent the score or the processes suggested by the composer on numerical, symbolic or graphic spaces has great importance. Changing the representation of an object permits one to see, to consider, to observe and finally to understand it in a different manner. The modeling process is transformed in a pragmatic analysis of the musical phenomena [14] leads us step by step to a model of Cage's thinking
Concerning the player, in pieces as different as *Solos* from the *Concert for Piano and Orchestra*, the player can concentrate on performing when using the CAP interface. After determination of the duration, he does not have to prepare his personal version, and will ignore completely what music he is going to perform. The player may be involved in determining the setting of the performance: relative density of the audio elements (length of silences), orienting the choice of the elements, using characterization of the material given (i.e. pitch, timbre, dynamics etc.) on a local or a global criteria. One might wonder: when all decisions regarding the order of the events of the scores and the timing etc. are made by a computer, what remains to be done by the performer/interpreter? Firstly, these calculations need to be defined by several parameters, which are personal choices. The result of the computing process is highly dependent of the interface choice and organization, therefore is part of the interpretation. Secondly, the performer can react and decide if and when an event is played, regardless of the fact that it is scheduled and displayed by the software. In this way, the computing result is a proposition that could be modified by the performer, which is in conformity with Cage's original instructions.

We are looking forward to provide a "Full" version of "Computer Assisted *Concert*", including a Computer "conductor" and 14 computers for players. Performers' interface with 3 different displaying modes, metadata on some sequencing events impossibilities, and expanding score generation models according to time constraints evolution of various "descriptors".

8. REFERENCES

[1] J. Cage, *Concert for piano and orchestra*, Peters by Henmar Press, 1960.

[2] B. Sluchin, "*Linaia Agon*, towards an interpretation based on the theory", *Proceedings of Iannis Xenakis International Symposium*, Athens, Greece, p. 299-311, 2005.

[3] P. Y. Bosseur, "John Cage Concert for Piano and Orchestra (1957–1958)", *Musiques Contemporaines Persepectives analytiques (1950-1985)*. Minerva, Paris, 2007.

[4] D. Nicholls, *John Cage*, University of Illinois Press, Unbana and Chicago, 2007.

[5] K. Gann, *No Such Thing as Silence: John Cage's 4'33*, Yale University Press, 2010.

[6] K. Boehmer, I. Pepper, "Chance as Ideology", *October,* vol. 82 IS, pp. 62-76, 1997.

[7] J. G. Chilton, *Non-intentional performance practice in John Cage's solo for sliding trombone*, DMA dissertation, University of British Columbia, 2007.

[8] J. Pritchett, *The Music of John Cage*, Cambridge University Press, Cambridge, 1993.

[9] D. Campana, *Form and Structure in the Music of John Cage*, Ph.D. Northwestern University Evanston 1985.

[10] R. Reynolds, J. Cage, "John Cage and Roger Reynolds: A Conversation", *Musical Quarterly*, Vol. 65, No. 4, pp. 573-594, 1979.

[11] J. Cage, R. Kostelanetz, "His Own Music", *Perspectives of New Music*, vol. 25, 1/2, 1987, pp. 88-106.

[12] A. A. Cournot, *Exposition de la théorie des chances et des probabilités*, T. 1, Hachette, Paris, 1843.

[13] I. Xenakis, *Musiques Formelles*, Stock Musique, Paris, 1981.

[14] D. Keller, B. Ferneyhough, "Analysis by modeling: Xenakis's ST/10 080262", *Journal of New Music Research*, 33(2), 161-171, 2004.

[15] James Pritchett, *The Music of John Cage*, New York: Cambridge University Press, 1993.

[16] David P. Miller, "Indeterminacy and Performance Practice in Cage's Variations", *American Music*, vol. 27, n° 1, 2009, pp. 60-86.

ADAPTIVE SPATIALIZATION AND SCRIPTING CAPABILITIES IN THE SPATIAL TRAJECTORY EDITOR HOLO-EDIT

Charles Bascou
GMEM
Centre National de Création Musicale
15 rue de Cassis
Marseille, France
charles.bascou@gmem.org

ABSTRACT

This paper presents recent works on controlling and editing sound spatialization on multiple speakers based on sound descriptors. It has been implemented as an extension of Holo-Edit, an OpenSoundControl compliant multitrack spatial trajectory editor developed at GMEM. An SDIF interface has been implemented allowing importing and visualizing sound descriptors generated by third party softwares. A set of scripting tools is proposed to process and map these time-tagged data to sound trajectory generation.

1. INTRODUCTION

Sound spatialization has become an important field in the past decades. From Karlheinz Stockhausen early experiments of moving acoustic sound sources around a set of microphones to contemporary cinematographic multichannel mastering, there has been a lot of growing interests in experimenting with sound diffusion across multiple loudspeakers. We can distinguish two main approaches in the work with sound in space. One has been the GRM's Acousmonium [1], initiated by Francois Bayle in 1974 which tends to spatialize stereo tracks manually from a mixing desk onto a loudspeaker orchestra. In its main principle, using an eclectic set of speakers allows the sound to be spatialized by itself, even with no intervention from the electroacoustic music performer. The acoustic characteristics of the different loudspeakers and their position in the concert hall makes them unique sound sources with their specific color and *timbre*, some enhancing high frequency, some medium ones, etc. Movements of sound are then created obviously by electroacoustic music interpretation but also by the *movement* and the *energy* of the sound itself. The other main approach has been the virtual acoustic model where sound spatialization is performed by mathematical and acoustical laws of sound in space. One is for example the distance cue which is simulated by attenuating sound volume (roll-off), filtering high frequencies (air absorption) and increasing the reverberation ratio of the spatialized sound. Here, movements of sound is done by controlling the virtual source position with calculated trajectories or via external input devices such as joystick or graphics tablet.

The focus of this paper is between these two models, in other words to get virtual acoustic movements of sound closer to the sound properties and behaviors. Our software environment, Holo-Edit, is a graphical editor for spatialization trajectories and is particularly adapted for controlling virtual acoustic DSP softwares. Our main goal here is to enhance trajectory editing and spatial cues by taking into account the inner structure and dynamic profiles of the sound to be spatialized. This is using the principle of adaptive audio effects [1] applied to sound spatialization. We will first detail the main features of the used environment Holo-Edit, then detail our motivations and finally present the proposed adaptive spatialization framework.

2. HOLO-EDIT FEATURES

Holo-Edit is initially part of the HoloPhon project initiated in 1996 by Laurent Pottier at GMEM [2] [2]. This project was focused on sound spatialization editing and control. Growing computing power allowed to develop custom DSP spatialization softwares written in MaxMSP, under the generic name Holo-Spat. For now, we will detail Holo-Edit features, as it is the main environment for our experiments.

2.1 Workflow

Holo-Edit is a standalone application written in Java/Jogl. The main underlying paradigm is the control of external DSP softwares via the OSC protocol. In [3], we showed the benefits of a stratified approach in sound spatialization environments. In this scheme, Holo-Edit has its place as an authoring tool for composing with space. All DSP processes are handled in other layers, e.g. in external softwares like MaxMSP, PureData, SuperCollider, etc. Holo-Edit only deals with movement of sound in space, using the timeline paradigm found in traditional DAWs to record, edit, and play back control data. Although a straightforward OSC protocol has been defined in order especially to

[1] Groupe de Recherches Musicales - Paris

[2] Groupe de Musique Experimentale de Marseille

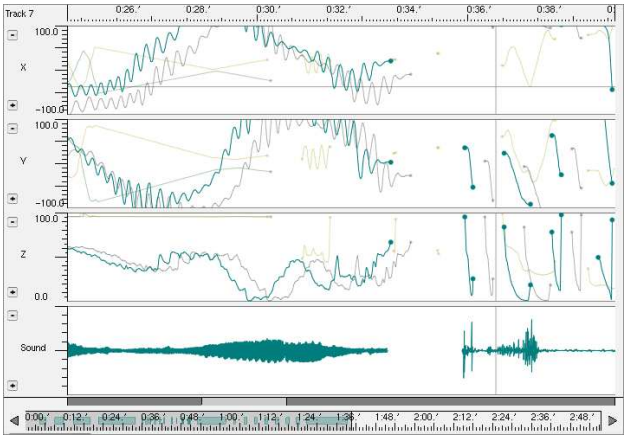


Figure 1. *Time Editor* view with cartesian xyz components and sound waveform.

keep Holo-Edit and the external spatialization software in sync in terms of time region selection and track muting and visualizing, an effort is made to standardize the way Holo-Edit deals with common control messages like source position. As it is the most promising attempt in that purpose, we are currently following the SpatDIF [4] initiative, with first experimental interfaces in the Jamoma Modular environment [5].

2.2 Multitrack Data representation and editing

Holo-Edit manipulates trajectory objects which is a set of time-tagged 3D points. These trajectories have an onset and offset time. Various graphical editing function can be achieved on these object like stretch, extend, trim, join. Maximum time resolution for the trajectories and points has been set to one millisecond allowing precise sound event/position mapping.

Additionally, audio waveforms can be imported from traditional sound files, and included in the composition. Sound cues are then triggered in parallel with the trajectories. It also helps in synchronizing sound events and corresponding position in space while editing.

Various representations of spatialization data are proposed. The *Room Editor* is a top-view editor of the virtual scene where you can move points and trajectories. The *Time Editor* in Figure 1 focus on time/data representation in a similar way as DAW softwares do for automation curves. The user can view and edit individually cartesian and polar coordinates components as curves. In this view, you can also visualize pre-imported sound waveforms time-aligned with corresponding 3d positions.

The *3d Room* shown in Figure 2 offers a 3d representation of the trajectories in the virtual scene although no editing can be achieved in it.

Holo-Edit is closely inspired by traditional Multitrack DAW. The *Score* view uses the timeline paradigm to represent the whole spatialized composition. Sound blocks and trajectory blocks can be moved and copied from one track to another. Traditional solo and mute functions are also implemented.

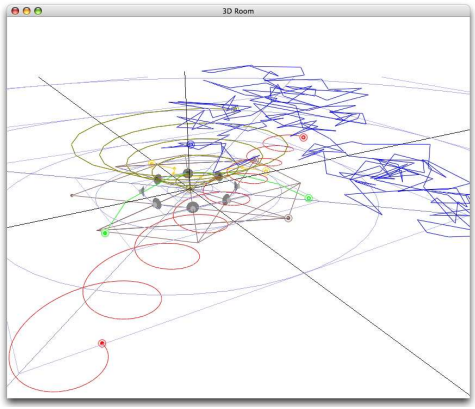


Figure 2. *3D Room* visualization.

2.3 Data Recording via OSC

Trajectories can be generated in various ways. One of them is the ability to record track position in real time from an external program or device via the OSC protocol. Several tracks can be recorded at the same time. For each track, the *segment* message allows to start a new trajectory where all future recorded points will be stored. It is particularly meaningful when using graphics tablet as input device, starting a new trajectory each time the stylus touches the tablet. In this purpose, user can import soundfiles and then, while playing them, record in real time their corresponding trajectories. All the proposed transformation functions can be applied then to smooth, scale or translate the recorded movements.

2.4 Generation/transformation plugins

In the Holo-Edit environment, the ability to process spatially and temporally the spatialization data is an important feature. Graphical editing could be a good help but sometime lacks accuracy for precise and repetitive tasks. A plugin interface has thus been defined for generative and transformative functions. They share a common scheme in their application. Functions are applied into the global time selection. They can output and/or apply the result on one, all or only visible tracks. It allows to make batch process on several tracks and trajectories. There are three process categories : *Generative Functions*, *Spatial Transformations* and *Temporal Transformations*. *Generative Functions* includes circular, lissajou, brownian and random algorithm. *Spatial Transformations* deals with basic geometrical transformation like rotation, translation, proportion. It also includes some more specific processes like exaggeration which scale the local movement of a trajectory leaving the main form unchanged. In the *Temporal Transformations*, the user can perform time stretch, acceleration or time reverse.

3. MOTIVATIONS

One often uses algorithmic functions to generate movements of a specified sound. In this framework, circular, brownian and random movements are the most commonly

used and offer, in their combinations, a wide range of different spatial figures. Though their efficiency, it is not so easy to tune their parameters to fit the spatialized sound behaviors, for example, finding the correct circular speed with an iterative sound (supposing quasi-synchronous iterations). The same problem comes when using random or brownian movements on chaotic granular sound materials where we would wish internal sound events to be placed individually. Quite often it results in ”spatial contradictions” that is when the spatialization movement contradict the sound internal behavior and *energy*. Note that this *energy* is quite a subjective and cultural notion. Instrumental sound is good illustration of this phenomenon. When listening to instrumental sound without any visual stimuli, the felt movement and energy are generally strongly associated with those engaged in the instrumental interpretation of a musician. String instrumental sound for example inspire the back and forth movement of the bow. In this framework, the main idea is to find in sound dynamic characteristics these inner profiles, such profiles helping then in setting the virtual movement in sync with its properties.

In a traditional sound synthesis workflow, it is not common to generate spatialization movements when synthesizing or mixing sound. In the purpose of generating/enhancing movements closely related to the sound inner structure, it is obvious that when working with sound synthesis, we could deduce meaningful time profiles from the different modulations and interactions defined in the synthesis process. Such time profiles could greatly help in creating spatial movements. Unfortunately, they are not so easy to route and store in an efficient way. If the internal control signals of these synthesizer are accessible, MIDI sequencers could be a solution but with a known lack of data and time accuracy.

Moreover, spatialization is still a process which is quite difficult to setup in a home studio environment. This work often takes place in dedicated spatialization studio. This is contributing in making sound generation/mixing and spatialization two processes which hardly come together in the same place and time.

The main idea is then to be able to analyze sound to be spatialized, extracting characteristic profiles in its structure. These profiles can then be mapped to various parameters of the trajectory generation/transformation.

4. ADAPTIVE FRAMEWORK

4.1 SDIF Data import and Visualization

SDIF (Sound Description Interchange Format) [6] is a standard generic, open, and multi platform format for sound description data storage. An SDIF file contains one or more sequences of entities called *frames*. Each Frames are time-tagged and typed among a wide range of defined sound descriptors. For example, let’s cite Fundamental Frequency, Loudness, Noisiness as traditional synchronous data flux associated with sounds. These descriptions can also be asynchronous like transient markers or chord separation markers. In this case data rate is not necessarily constant.

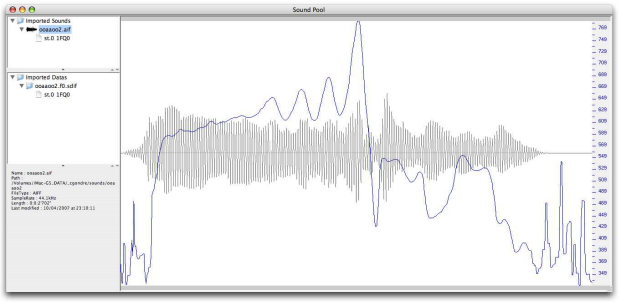


Figure 3. Waveform and its fundamental frequency shown in the *SoundPool* window.

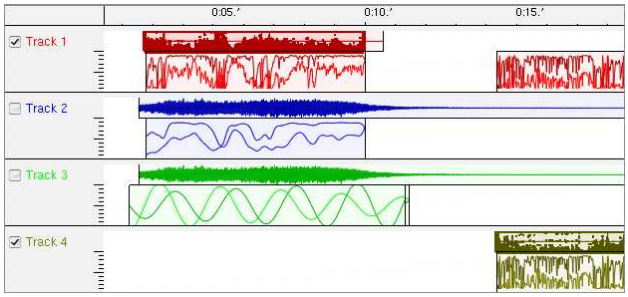


Figure 4. *Score* window with waveform and data visualization on top of the trajectories.

We chose SDIF as it is widely adopted in sound analysis softwares like IRCAM AudioSculpt and AsAnnotation or Michael Klingbeil’s SPEAR [7]. Current SDIF implementation includes a library in C/C++. Since Holo-Edit has been written in java, we had to find a way for interfacing with this standard SDIF library. Thanks to the SWIG interface (Simplified Wrapper and Interface Generator) [8], we could develop a native java wrapper for SDIF. It is composed of a java classes and a platform specific jnilib.

SDIF import in Holo-Edit is done via the *SoundPool* window. As shown in Figure 3, one can overlay sdif data and the corresponding previously imported waveform. These data objects can then be dropped in the *Score* window on a specific track. At this step, these objects are not yet interacting with the spatialization score. They act more as visual cues and markers for enhanced editing especially in the *TimeEditor* window where precise resynchronization of trajectories and points can be achieved.

An important feature is that data objects can be linked with waveform objects so that they are always in sync in the *Score* and the *TimeEditor* window. It is particularly useful when using these data to generate or modify trajectories as we will see in the next section.

4.2 Direct Mapping Plugins

One key feature of the SDIF support is the ability to use these time profiles to generate or transform trajectories. A first step has been to propose a simple plugin interface that converts these time-tagged values into 3d point parameter.

This generative function works in the same scheme as the others functions do. It uses the global time selection to define the region where the new trajectory will be written.

It generates trajectory in the selected tracks. Data access is also done according to the time selection. The plugins scans the score for each overlapping data object instances in the score and then construct a menu on the interface for selection of the data we want to use. A special naming protocol has been defined referencing a data object instance characterized by the SDIF filename, the SDIF stream id, its begin time in the score and its parent track. For example, an fundamental frequency data object instance on track 2 starting at 52 seconds would be named :

```
3-francesca_04.f0.sdif - st.0 1FQ0 - begin time
    =0:0:52:000 - Track:2
```

Note that, as these data objects are in a kind of global scope, the plugin can access them from all tracks. That can avoid duplicating the same data object on multiple tracks when the user wants to generate different mapping of the same data on multiple tracks.

In the cartesian coordinates version of the plugin, the user can assign a different data instance to each XYZ components. As there can be multiple streams of time tagged value in a data object, user has to choose an available one depending on the analysis software which generated the SDIF file. For Example, with fundamental frequency data object (FQ0), AudioSculpt gives us four streams which are *Frequency* in Hz, *Confidence* coefficient, *Score* coefficient and linear *RealAmplitude*. All these values have different unit so a scaling process has to be made. This is done automatically with linear mapping from minimum and maximum of the data values to -1. and 1. for cartesian components. In fact, scaling and translating can be made afterwards with all the graphical and algorithmic functions already available so there is no need to propose scaling feature in the plugin interface.

This direct data mapping can be interesting for simple experiments. It allows to easily transform data points into trajectory points. These profiles can then be shaped in space and time with all the available editing functions like scaling, translation, rotation, etc. But as we will see in the next section, it becomes quite limiting when using more complex data (like multidimensional ones) or experimenting original transformative plugins.

4.3 Script interface

The direct mapping plugins presented in the last section can be really helpful for simple data. But quickly comes the need to have more complex mapping of these profiles. Obviously, in the scheme of adaptive spatialization, this process is central; this is where the user defines the relation between sound characteristics and spatial cues. As it is a very vast field, it is important to let the user the ability to experiment by himself original relations. We thus had to find a workflow as open as possible with a rich expressivity that can well define complex relations and this is what scripting can do. Notably, scripts may be particularly useful in different cases:

1. Performing repetitive tasks efficiently.
2. Applying algorithms with precision on some data.
3. Getting some information about available data.
4. Mathematical functions availability.
5. Creating libraries for trajectory transformations and generations.
6. Algorithms fast experimentations.

Thanks to the *Groovy* project [9] which aims to propose an agile and dynamic language for the Java Virtual Machine, we could build a script interface integrated in the internal Holo-Edit environment. Groovy is built on top of Java and is found to be an easy to learn and powerful object oriented scripting language. Its benefits are multiple : on the fly execution of code from a text buffer or from interactive console, dynamic typing, Java context integration, and so on. The last one is particularly interesting as it easily allows to access to any java classes and functions the main java program uses. In our case, it has greatly simplified the interface with the Holo-Edit Java objects such as 3d points, trajectories and tracks. Moreover an abstract java class has been developed to facilitate scores data access and scripting with them.

The *Script* window presents a text area where the script can be edited. An additional text area is shown called ”Values from score” where useful local variables are proposed. They are *begin*, *end* and *duration* of the global time selection when the script function was called. It also present the name of the available SDIF data instances overlapping the time selection. This name can be copy and paste in the script and is used as a reference for data instances. Getting data instance handle is made for example with :

```
mySDIFdata = getSDIFdata("quat-cel1.f0.sdif -
     st.0 1FQ0 - begin time=0:0:38’213 -
    Track:0")
```

Some helper functions has been included in the interface especially for sampling data at a specific time making transformative script easier when point timestamps of the transformed trajectory have to remain unchanged.

Here is a simple example of transformative script where fundamental frequency is mapped to the z components of the trajectory points :

```
import holoedit.data.*;

f0data = getSDIFdata(gp,
  "3-francesca_04.f0.sdif - st.0
  1FQ0 - begin time=0:0:52’290 - Track:2")
int dateBegin = 5171;
int dateEnd = 74636;
double dur = 69465;

float min = minFieldValue(f0data,0);
float max = maxFieldValue(f0data,0);
float mean = meanFieldValue(f0data,0);
float range = max - min;

HoloPoint point;
int date;

for (int i = 0; i < gp.copyTrack.size(); i++)
{
  point = gp.copyTrack.elementAt(i);
  date = (point.date - dateBegin)*10;
  if( hasDataAtTime(f0data,date))
    point.z = 50 + ( getDataAtTimeField(f0data,
        date,0) -mean ) * 200 / range;
}
```

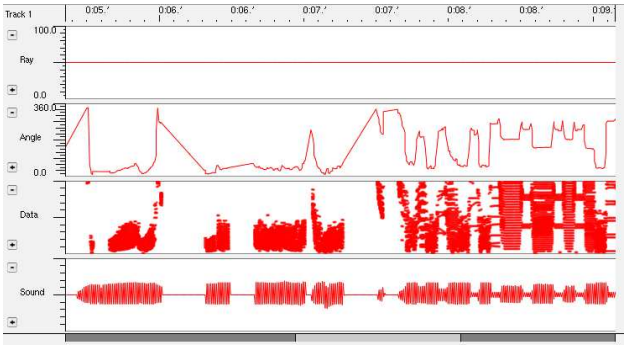


Figure 5. *Time Editor* view of the trajectory generated by brightness mapping.

Another benefit of using script is that the user can generate or transform a trajectory from multiple data instances. For example, he could use voiced/unvoiced segmentation and fundamental frequency to define rules of spatial behaviors; let's say for example random movement for unvoiced part and frequency dependent position for voiced part.

4.4 User case: sound brightness mapping

We will present in this section an example of application of this adaptive framework. We chose to use sound brightness as it has found to be very efficient to discriminate sound events in a complex mix. Its profile contains a lot of relevant informations as it detect timbre abrupt changes. It could be compared to fundamental frequency but with more robust results in a great variety of sound materials (e.g. non harmonic sounds or complex mixture). An interesting feature of sound brightness mapping is that its profile contains good representation of transients and steady parts of the sound.

Moreover, this example is good illustration of data filtering and processing. Indeed, no known end-user software can produce brightness, or spectral centroid its DSP equivalent, SDIF files. We thus had to compute it directly from spectral data. In that purpose, we used a spectral peak analysis to preprocess and simplify the raw spectrogram. The peak analysis outputs for each frame a set of peaks defined by its frequency, amplitude and phase. So we included in the mapping script the computation of an estimate of the spectral centroid, that is the frequency barycenter of the peak set. It gives us a one dimensional data array easily mappable to a trajectory parameter.

The experiment has been to map this value to the azimuth of the source, the distance remaining constant. The data rate, and consequently the point rate, has been set to 5 ms to be able to capture fast timbre changes. The result is kind of spatial magnification of the sound. Each occurring similar event gets its own position in space giving a very coherent spatial image of the sound. The Figure 5 shows the resulting trajectory in the Time Editor with ray, azimuth, data and waveform curves. This technique works great for complex mixture or sound material but harmonic instrumental sounds get poorer results as their *timbre* evolve much slowly. In this case, it should be probably better to elaborate algorithms based on fundamental frequency.

5. CONCLUSIONS AND FUTURE WORK

Adaptive spatialization offers a great field of new possibilities in trajectory generation and transformation. We proposed, to exploit it, an experiment interface in end-user spatialization authoring tool. It has the benefit to allow original editing functions while keeping fast and easy graphical fine tuning of spatial behaviors. We planned some enhancements of the script interface to get to a more global scope on the spatialization score. The generation of trajectories on multiple track for example could help when generating parallel hardly correlated trajectories. Some particular generation and transformation algorithms, especially the one based on sound brightness, will also be implemented has standard Holo-Edit plugins in order to make them available to user impervious to script editing. The Holo-Edit software and sources are available for download on the GMEM website `http://www.gmem.org`.

6. ACKNOWLEDGMENTS

The author would like to specially thank Leopold Frey and Charles Gondre for their great contributions to the development of this project and composer Rodrigo Cicchelli Velloso for his interest in the musical application of the prototyped framework.

7. REFERENCES

[1] X. Amatriain, J. Bonada, A. Loscos, and J. Arcos, "Content-based transformations," in *Journal of New Music Research, 32(1)*, pp. 95–114, 2003.

[2] L. Pottier, "Dynamical spatialization of sound. holophon : a graphic and algorithmic editor for sigma1," in *Dafx98 Proceedings*, 1998.

[3] N. Peters, T. Lossius, J. Schacher, P. Baltazar, C. Bascou, and T. Place, "A stratified approach for sound spatialization," in *Proceedings of The 6th Sound and Music Computing Conference*, 2009.

[4] N. Peters, "Proposing spatdif - the spatial sound description interchange format," in *Proceedings of the 2008 International Computer Music Conference, Belfast*, 2008.

[5] T. Place and T. Lossius, "Jamoma: A modular standard for structuring patches in max," in *Proceeding of the International Computer Music Conference 2006*, 2006.

[6] D.Schwarz and M. Wright, "Extensions and applications of the sdif sound description interchange format," in *Proceedings of the International Computer Music Conference, Berlin*, 2000.

[7] M. Klingbeil, "Software for spectral analysis, editing, and synthesis," in *Proceedings of the International Computer Music Conference, Barcelona*, 2005.

[8] *SWIG*. `http://www.swig.org/`.

[9] *Groovy*. `http://groovy.codehaus.org/`.

ON ARCHITECTURE AND FORMALISMS FOR COMPUTER-ASSISTED IMPROVISATION

Fivos Maniatakos * ** Gerard Assayag * Frederic Bevilacqua ** Carlos Agon*
* Music Representation group (RepMus)
** Real-Time Musical Interactions team (IMTR)
IRCAM, UMR-STMS 9912
Name.Surname@ircam.fr

ABSTRACT

Modeling of musical style and stylistic re-injection strategies based on the recombination of learned material have already been elaborated in music machine improvisation systems. Case studies have shown that content-dependant regeneration strategies have great potential for a broad and innovative sound rendering. We are interested in the study of the principles under which stylistic reinjection could be sufficiently controlled, in other words, a framework that would permit the person behind the computer to guide the machine improvisation process under a certain logic. In this paper we analyze this three party interaction scheme among the isntrument player, the computer and the computer user. We propose a modular architecture for Computer Assisted Improvisation (CAO). We express stylistic reinjection and music sequence scheduling concepts under a formalism based on graph theory. With the help of these formalisms we then study a number problems concerning temporal and qualitative control of pattern generation by stylistic re-injection.

1. INTRODUCTION

New computer technologies and enhanced computation capabilities have brought a new era in real-time computer music systems. It is interesting to see how artificial intelligence (AI) technology has interacted with such systems, from the early beginning until now, and the effect that these enhancements have had when setting the expectations for the future. In the 2002's review paper [1], computer music systems are organized in three major categories: (1) Compositional, (2) Improvisational and (3) Performance systems. Concerning the limits between what we call computer improvisation and computer performance, the authors of [1] state the following:

"... Although it is true that the most fundamental characteristic of improvisation is the spontaneous, real-time, creation of a melody, it is also true that interactivity was not intended in these approaches, but nevertheless, they could generate very interesting improvisations."

According to [1], first improvisation systems did not address directly interactivity issues. Algorithmic composition of melodies, adapatation in harmonic background, stochastic processes, genetic co-evolution, dynamic systems, chaotic algorithms, machine learning and natural language processing techniques constitute a part of approaches that one can find in literature about machine improvisation. However, most of these machine improvisation systems, even with interesting sound results either in a pre-defined music style or in the form of free-style computer synthesis, did not address directly the part of interaction with human.

During the last years, achievements in Artificial Intelligence and new computer technologies have brought a new era in real-time computer improvisation music systems. Since real-time systems for music have provided the framework to host improvisation systems, new possibilities have emerged concerning expressiveness of computer improvisation, real-time control of improvisation parameters and interaction with human. These have resulted to novel improvisation systems that establish a more sophisticated communication between the machine and the instrument player. Some systems have gone even further in terms of interactivity by envisaging a small participation role for the computer user.

However, as for the interaction design of many of these systems, the role of the computer user in the overall interaction scheme often seems to be neglected, sometimes even ignored. What is the real degree of communication that machines achieve to establish with the instrument players in a real-world improvisation environment? Is the study of bipartite communication between instrument player - computer sufficient enough to model real-world complex improvisation schemes? More importantly, do modern computer improvisation system really manage to exploit the potential of this new form of interactivity, that is, between the computer and its user? And what would be the theoretical framework for such an approach that would permit a double form of interactivity between (1) computer and instrument player (2) computer and its user?

In our research we are concerned with the aspects of enhanced interactivity that can exist in the instrument *instrument player - computer - user* music improvisation framework. We are interested particularly on the computer - user communication channel and on the definition of a theoretical as well as of a the computational framework that can permit such a type of interaction to take place in real-time. Such a framework differs from the conventional approach

of other frameworks for machine improvisation in that the computer user takes an active role in the improvisation process. We call the context we study *Computer Assisted Improvisation (CAI)*, due to the shift of the subject role from the computer to the computer user.

Later in this report, we propose a model for three-party interaction in collective improvisation and present an architecture for Computer Assisted Improvisation. We then introduce a formalism based on graph theory in order to express concepts within computer assisted improvisation. This formalism succeeds in expressing unsupervised, content-dependant learning methods as well as the concept of "stylistic re-injection for the pattern regeneration, furthered enriched with new features for user control and expressivity. Through a bottom-up approach we analyze real-world sequence scheduling problems within CAI and study their resolvability in terms of space and time complexity. In this scope we present a number of real world music improvisation problems and show how these problems can be expressed and studied in terms of graph theory. Finally we give notions about our implementation framework GrAIPE (Graph assisted Interactive Performance Environment) in the real-time system Max-MSP, as well as about future research plans.

2. BACKGROUND

One of the first models for machine improvisation appeared in 1992 under the name *Flavors Band* [4]. Flavors Band a procedural language for specifying jazz and procedural music styles. Despite the off-line character of the approach, due to which one could easily classify the system to the computer-assisted composition domain, the advanced options for musical content variation and phrase generation combined with musically interesting results inspired later systems of machine improvisation. In the computer assisted improvisation context, the interaction paradigm proposed by Flavors Band could be regarded as a contribution of major importance, due to the fact that it assigns the computer user with the role of the musician who takes high-level offline improvisation decisions according to machine's proposals. Adopting a different philosophy as for the approach, GenJam [6] introduced the use of genetic algorithms for machine improvisation, thus being the 'father' of a whole family of improvisation systems in the cadre of evolutionary computer music. Such systems make use of evolutionary algorithms in order to create new solos, melodies and chord successions. It was also one of the first systems for music companion during performance. Evolutionary algorithms in non supervised methods were introduced by Papadopoulos and Wiggins in 1998 [9]. More recent learning methods include reccurent neural networks [5], reinforcement learning [11], or other learning techniques such as ATNs (Augmented Transition Networks) [7] and Variable order Markov Chains [12].

Thom with her BAND-OUT-OF-A-BOX(BOB) system [2] addresses the problem of real-time interactive improvisation between BOB and a human player. Sever years later, the LAM (Live Algorithms for Music) Network's manifesto underlines the importance of interactivity, under the term *autonomy* which should substitute the one of *automation*. In [14] the LAM manifesto authors describe the Swarm Music/Granulator systems, which implements a model of interactivity derived from the organization of social insects. They give the term 'reflex systems' to systems where 'incoming sound or data is analysed by software and a resultant reaction (e.g. a new sound event) is determined by pre-arranged processes.' They further claim that this kind of interaction is 'weakly interactive because there is only an illusion of integrated performer-machine interaction, feigned by the designer'. With their work, inspired by the animal interaction paradigm, they make an interesting approach to what has prevailed to mean 'human - computer interaction' and bring out the weak point inside real-time music systems' design. However, even if they provide the computer user with a supervising role for the improvisation of the computer, they don't give more evidence about how a three party interaction among real musician - machine - computer user could take place. Merely due to the fact that they consider their machine autonomous. But if the human factor is important for a performance, this should refer not only to the instrument player but to the person behind the computer as well. At this point, it seems necessary to study thoroughly the emerging three party interaction context where each of the participants has an independent and at the same time collaborative role.

Computer - computer user interaction is studied instead in the framework of live electronics and live coding, under the 'laptop-as-instrument' paradigm. In [15], the author describes this new form expression in computer music and the challenges of coding music on the fly within some established language for computer music or within a custom script language. The most widespread are probably the aforementioned SuperCollider [19], a Smalltalk derived language with C-like syntax, and most recently ChucK, a concurrent threading language specifically devised to enable on-the-fly programming [20]. Live coding is a completely new discipline of study, not only for music but also for computer science, phsycology and Human Computer Interaction as well. Thus, it seems -for the moment- that live coding is constrained by the expressivity limitations of the existing computer languages, and that it finds it difficult to generalize to a more complicated interaction paradigm which could also involve musicians with physical instruments.

2.1 OMax and stylistic reinjection

An interaction paradigm which is of major interest for our research is this of the stylistic reinjection, employed by the OMax system [3]. OMax is a real-time improvisation system which use on the fly *stylistic learning* methods in order to capture the playing style of the instrument player. Omax is an *unsupervised, context-dependant* performing environment, the last stating that it does not have pre-accuired knowledge but builds its knowledge network on the fly according to the special features of the player's performance. The capacity of OMax to adapt easily to different music styles without any preparation, together with its ability to treat audio directly as an input through the

employment of efficient pitch tracking algorithms, make OMax a very attractive environment for computer improvisation.

It is worth pointing out that style capturing is made with the help of the Factor Oracle Incremental Algorithm, intruduced by Allauzen and al. in [13], which repeatedly adds knowledge in an automaton called Factor Oracle (FO). Concerning the generation process, this is based on FO and is extensively described in [3]. With the help of forest of Suffix Link Trees (SLTs) it is possible to estimate a Reward Function in order to find interconnections between repeated patterns, which is the key for the pattern generation. Through this method, one can construct a model that continuously navigates within an FO. By balancing linear navigation with pivots and adding a little of nondeterminism to the Selection Function decisions, the system can navigate for ever by smoothly recombining existing patterns. Due to the short-term memory effect, this recombination is humanly perceived as a generation, which, more importantly, preserves the stylistic properties of the original sequence. The creators of the system call this method *stylistic reinjection*: this method relies on recollecting information from the past and re-injecting it to the future under a form which is consistent to the style of the original performance. Based on this principle and employing further heuristic and signal processing techniques to refine accordingly selections and sound rendering, OMax succeeds musically in a collective, real-world improvisation context.

Finally, OMax provides a role for the computer user as well. During a performance, user can change on the fly a number improvisation parameters, such as the proportion between linear navigation and pivot transitions, the quality of transitions according to common context and rhythm similarity, define the area of improvisation and cancel immediately the system's event scheduling tasks in order to access directly a particular event of the performance and reinitiate all navigation strategies.

2.2 The Continuator

An other system worth mentioning is The Continuator [12]. Based on variable-order Markov models, the system's purpose was to fill the gap between interactive music systems and music imitation systems. The system, handling except for pitch also polyphony and rhythm, provides content-dependent pattern generation in real-time. Tests with jazz players, as well with amateurs and children showed that it was a system of major importance for the instrument player - computer interaction scheme.

3. MOTIVATION

In this point, it is interesting to have a look over interaction aspects between the musician and the machine. Maintaining style properties assures that similar type of information travels in both directions throughout the musician - machine communication channel, which is a very important prerequisite for interaction. Moreover, diversity in musical language between machines and physical instrument playing, is one of the main problems encountered by evolutionary improvisation systems. OMax deals well with this inconvenience, in the sense that the patterns being regenerated and the articulation within time is based on the music material played the performer. However, when for interaction one should study not only information streams but also design of modules responsible for the *Interpretation* of received information [14]. In the case of the instrument player, human improvisors can interpret information according to their skills developed by practicing with the instrument, previous improvisational encounters and stylistic influences. These skills allow -or not- reacting to surprise events, for instance a sudden change of context. Concerning the machine, OMax disposes an interpretation scheme that stores information in the form FO representation. The question that arises concerns the capability of this scheme to handle surprise. This question can be generalized as follows: Can stylistic reinjection approach adapt its pre-declared generation strategies to short-time memory events? The last implies the need for an specifically configured autonomous agent capable of detecting short-time memory features of a collective improvisation, understanding their musical nuance and transmitting information to the central module of strategy generation.

Concerning stylistic reinjection, this approach currently permits a several amount of control of the overall process. We described in previous section the computer user's role in the OMax improvisation scenario. However, It seems intriguing to investigate further the role the computer user can have in such a scenario. For instance, wouldn't it be interesting if the computer user could decide himself the basic features of a pattern? Or if he could schedule a smooth passage in order that the computer arrives in a specific sound event within a certain time or exactly at a time?

We believe that the study of three-party interaction among can be beneficial for machine improvisation. In particular, we are interested in the 'forgotten' part of computer - computer user interaction, for which we are looking forward to establishing a continuous dialog interaction scheme. Our approach is inspired from the importance of the human factor in a collective performance between humans and machines, where humans are either implicitly (musicians) or explicitly (computer users) interacting with the machines. Though this three party interaction scheme, computer user is regarded as an equal participant to improvisation.

With respect to existing systems, our interest is to enhance the role of the computer user from the one of supervisor to the one of performer. In this scope, instead of controlling only low level parameters of computer 's improvisation, the user is set responsible of providing the computer with musical, expressive information concerning the structure and evolution of the improvisation. Inversely, the computer has a double role: first, the one of an augmented instrument, which understands the language of the performer and can regenerate patterns in a low level and articulate phrases coherent to user's expressive instructions, and second, the one of an independent improvisor, able to respond reflexively to specific changes of music context or to conduct an improvisation process with respect to a pre-

specified plan.

Our work consists of setting the computational framework that would permit such operations to take place. This includes:1) the study of the interaction scheme, 2) the architecture for such an interaction scheme 3) an universal representation of information among the different participants and 4) a formalism to express and study specific musical problems, their complexity, and algorithms for their solution.

4. ARCHITECTURE

In this section we analyze three-party interaction in CAI and propose a computational architecture for this interaction scheme.

4.1 Three party interaction scheme

In figure 1, one can see the basic concepts of three party interaction in CAI. The participants in this scheme are three: the musician, the machine (computer) and the performer. Communication among the three is achieved either directly, such as the one between the performer and the computer, or indirectly through the common improvisation sound field. The term sound field stands for the mixed sound product of of all improvisers together. During an improvisation session, both the musician and the performer receive a continuous stream of musical information, consisting of a melange of sounds coming from all sources and thrown in the shared sound field canvas. They manage to interpret incoming information through human perception mechanisms. This interpretation includes the separation of the sound streams and the construction of an abstract internal representation inside the human brain about the low and high level parameters of each musical flux as well as the dynamic features of the collective improvisation. During session, the musician is in a constant loop with the machine. He listens to its progressions and responds accordingly. The short-time memory human mechanisms provides her/him with the capacity to continuously adapt to the improvisation decisions of the machine and the evolution of the musical event as a group, as well as with the ability to react to a sudden change of context.

On the same time, the machine is listening to the musician and constructs a representation about what he has played. This is one of the standard principles for human machine improvisation. Furthermore, machine potentially adapts to a mid-term memory properties of musician's playing, thus reinjecting stylistically coherent patterns. During all these partial interaction schemes, the performer behind the computer, as a human participant, is also capable of receiving and analyzing mixed-source musical information, separating sources and observing the overall evolution.

The novelty of our architecture relies mainly on the concept behind performer - machine's communication. Instead of restricting the performer's role to a set of decisions to take, our approach aims to subject the performer in a permanent dialog with the computer. In other words, instead of taking decisions, the performer 'discusses' his intentions with the computer. First, he expresses his intentions to the system under the form of musical constraints. These constraints concern time, dynamics, articulation and other musical parameters and are set either statically or dynamically. As a response, the computer proposes certain solutions to the user, often after accomplishing complex calculi. The last evaluates the computer's response and either makes a decision or launches a new query to the machine. Then the machine has either to execute performer's decision or to respond to the new query. This procedure runs permanently and controls the improvisation. An other important concept in our architecture concerns the computer's understanding of the common improvisation sound field. This necessity arises from the fact that despite for computer's ability to 'learn', in a certain degree, the stylistic features of the musician's playing, the last does not stand for the understanding of the overall improvisation. Thus, There has to be instead a dedicate mechanism that assures interaction between the machine and the collective music improvisation. Moreover, such a mechanism can be beneficial for the performer - machine interaction as well, as it can make the computer more 'intelligent' in his dialog with the performer.

4.2 General architecture for Computer Assisted Improvisation

However, for the conception of an architecture for CAI that permits three party interaction, there are a couple of important issues to take into account. First, the fact that the proposed interaction scheme is gravely constrained in time, due to the fact that all dialogs and decisions are to be taken in real time (though in the soft sense). Second, the computer should be clever enough to provide *high-level, expressive* information to the performer about the improvisation, as well as high level decision making tools.

The internal architecture for the computer system is shown in figure 2. This architecture consist mainly of six modules which can act either concurrently or sequentially. On the far left we consider the musician, who feeds information to two modules: the pre-processing module and the short-term memory module. The pre-processing module is responsible for the symbolic encoding of audio information and stylistic learning. On the far right part of the figure we see the renderer, the unit that sends audio information to the collective sound field.

The short-memory processing module serves the understanding of the short-term memory features of collective improvisation. In order to reconstruct internally a complete image for the improvisation's momentum, this modules gathers information both from the representation module and the scheduler; the first in order to know what is being played by the musician and the second for monitoring computer's playing in short-term. It is possible that in the future it will be needed that the short-term memory processing module will also include an independent audio and pitch tracking pre-processor in order to reduce the portion of time required for the detection of surprise events.

In the low-center part of figure 2 one can find the interaction core module. This core consists of a part that is responsible for interfacing with the performer and a *solver* that responds to his questions. The performer lances queries

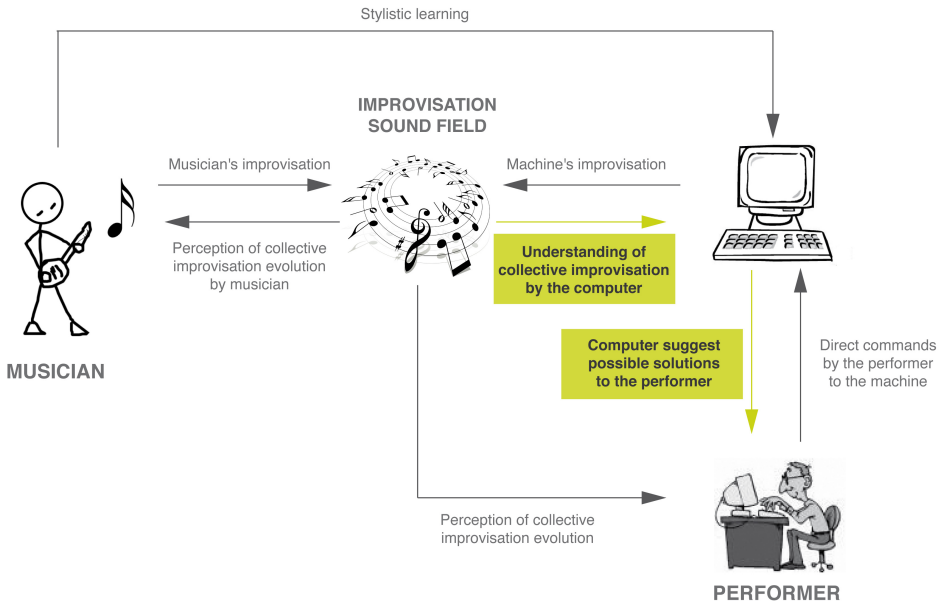


Figure 1. Three-party interaction scheme in Computer Assisted Improvisation. In frames (yellow) the new concepts introduced by the proposed architecture with respect to conventional interaction schemes for machine improvisation.

under the form of constraints. In order to respond, the solver attires information from the representation module. The time the performer takes a decision, information is transmitted to the scheduler. Scheduler is an intelligent module that accommodates commands arriving from different sources. For instance, a change-of-strategy command by the performer arrives via to the interaction core module to the scheduler.

The scheduler is responsible for examining what was supposed to schedule according to the previous strategy and organizes a smooth transition between the former and the current strategy. Sometimes, when contradictory decisions nest inside the scheduler, the last may commit a call to core's solver unit in order to take a final decision. It is worth mentioning that the dotted-line arrow from the short-term memory processing module towards the scheduler introduces the aspect of *reactivity* of the system in emergency situations: when the first detects a surprise event, instead of transmitting information via the representation module -and thus not make it accessible unless information reaches the interaction core-, it reflects information directly to the scheduler with the form of a 'scheduling alarm'. Hence, via this configuration, we leave open in our architecture the option that the system takes over *autonomous* action under certain criteria. The last, in combination with those mentioned before in this section establishes full three party interaction in a CAI context.

5. FORMALISMS FOR COMPUTER ASSISTED IMPROVISATION WITH THE HELP OF GRAPH THEORY

Further in this section, we address music sequence scheduling and sequence matching and alignment, two major problems for CAI. After a short introduction, we give formalisms for such problems with the help of graph theory.

5.1 Music sequence scheduling

Concerning the notion of *music scheduling* is usually found in literature as the problem of assigning music events to a particular time in the future, commonly within a real-time system [16]. Scheduling of musical events is one of the main functionalities in a real-time system [17] where the user is given the opportunity to plan the execution of a set of calculi or DSP events, in a relative or absolute manner, sporadically or sequentially. In a parallel study of the process of scheduling in music computing and other domains of research such as computer science and production planning, we could reason that for the general case described above, musical scheduling refers to the single machine scheduling and not the job-shop case.

We define *Music Sequence Scheduling* as the special task of building a sequence of musical tasks and assigning their execution to consecutive temporal moments.

In our study, we are interested to music sequence scheduling in order to reconstruct new musical phrases based on symbolically represented music material. Our objective is to conceptualize methods which will help the user define some key elements for these phrases, as well as a set of general or specific rules, and to leave the building and scheduling of the musical phrase to the computer. In an improvisation context, the last allows the performer taking crucial decisions in a high level; on the same time, the computer takes into account performer's intentions, sets up the low-level details of the phrase generation coherently to the user choices and outputs the relevant musical stream. For instance, a simple sequence scheduling prob-

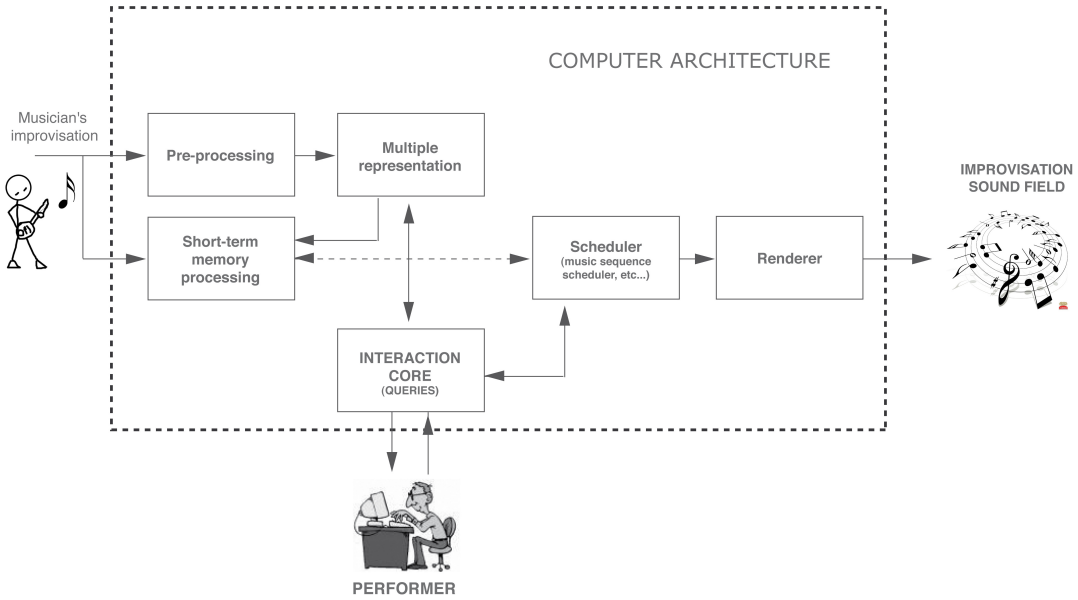


Figure 2. Overal Computer Architecture for CAI

lem would consist of navigating throughout a FO with the same heuristic such as the one used by OMax system, under the additional constraint that we would like to reach a particular state x in a particular moment t.

5.2 Music sequence matching and alignement

Music sequence matching concerns the capacity of the system to recognize if sequence is within its known material. Music sequence alignment is the problem of finding sequences within its corpus which are 'close' to a sequence of reference. The last pre-assumes the definition of a distance function according to certain criteria. Both are not in the scope of this paper, but are mentioned for reasons of clarity.

5.3 Formalisms

The structures used for learning in existing improvisation systems, even if effective for navigation under certain heuristics, are too specialized to express complex problem of navigation under constraints. For instance, a FO automaton is sufficient when for agnostic navigation under certain heuristics among its states, but fails to answer to problems of scheduling a specific path in time. In order to be able to express diverge problems of music sequence scheduling, alignment or more complex problems, we are obliged to use more general graph structures than the ones used in content-dependent improvisation systems. The advantage of this approach is that our research then can be then generalized to include other graph-like structures. Our method focuses on regenerating material by navigating throughout graph structures representing the corpus. Due to the reasons mentioned before we will express all stylistic reinjection related problems under the graph theory formalism.

Formally we describe the process of music sequence scheduling in the context of stylistic reinjection as follows:

Consider now a continuous sound sequence S. Suppose that for this sequence it is possible to use an adaptive segmentation method to segment S in n chunks according to content and a metric $m = f(d)$, $d \in D$, where D a set of descriptors for S. Each m causes different segmentation properties i.e different time analysis t_m. A symbolic representation of S would then be $S_m(t_m)$, $1 \leq m \leq M$, where M the number of metrics and $t_m = 0, 1, .., n_m \; \forall m \in M$.

Axiom 1 The musical style of a continuous sound sequence S can be represented by a countably infinite set P with $|P| \neq 0$ of connected graphs.

Definition 1

We define *stylistic learning* as a countably infinite set $F = \{Sl_1, Sl_2, .., Sl_n\}$, $|F| \neq 0$ of mapping functions $Sl_i : \mathbf{S}_m \rightarrow G_i(V_i, E_i)$, where $\mathbf{S}_m = S_{mt_m} \forall t_m \in [0, n_m]$ of sequence S for a metric m, $1 \leq m \leq M$, G_i a connected graph with a finite set of vertices V_i and edges E_i as a binary relation on V_i.

Definition 2 We define as *stylistic representation* the countably infinite set $P = \{G_i(V_i, E_i) : i \neq 0\}$ of digraphs.

Definition 3 We define as *sequence reinjection* a selection function
$R_{seq} : (G_i, q) \rightarrow \mathbf{S}_m$ with $R_{seq}(G_i, q) = \mathbf{S'_m}$ and $S'_{mt_m} = S_{mt'_m}$, q a number of constraints $q = h(m)$, $m \in (1, M)$.

Definition 4 We define as *musical sequence scheduling* as a scheduling function $R_{sch} : (R_{seq}, T_s(R_{seq})) \rightarrow \mathbf{S}_m$, with T_s the setup time for the sequence reinjection R_{seq}.

With these formalisms we can now begin to study stylistic representation, learning and sequence reinjection with the help of graphs. These issues now reduce to problems of constructing graphs, refining arc weights and navigating along the graphs under certain constraints.

In section 4.2 we underlined the importance of the short-term memory processing module. Even while the standard functionality, it should be employed with the mechanism to quickly decode information that has lately been added

to the representation, make comparisons with earlier added performance events and find similarities.

Definition 5 We define *Music Sequence Matching (MSM)* as a matching function $M : (\mathbf{S}_m, \mathbf{S'}_m) \rightarrow [0,1]$, where $\mathbf{S}_m$, $\mathbf{S'}_m$ symbolic representations of sequences S and S' for the same metric m.

Definition 6 We define *Music Sequence Alignment (MSA)* the alignment function $A : (\mathbf{S}_m, \mathbf{S'}_m, q) \rightarrow R$ where

$$A(\mathbf{S}_m, \mathbf{S'}_m, q) = \min\{\sum_q c_q x_q\}, \quad (1)$$

$\mathbf{S}_m, \mathbf{S'}_m$ symbolic representations of sequences S and S' respectively for the same metric m, q a number of string operations, $c_q \in \mathbb{R}$ a coefficient for an operation q and $x_q \in \mathbb{Z}$ the number of occurrences of operation q.

With the help of the previous definitions we are ready to cope with specific musical problems.

6. A SIMPLE PROBLEM ON STYLISTIC REINJECTION IN CAI

Problem

Let a musical sequence S symbolically represented as S_n, $n \in [0,N]$, and $s,t \in [0,N]$ a starting and target point somewhere within this sequence. Starting from point s, navigate the quickest possible until the target t, while respecting sequence's stylistic properties.

Definition 7 With the help of axiom 1 and definitions 1, 2, we apply stylistic learning and create a stylistic representation digraph $G(V,E)$ for S_n with set of vertices $V = \{0,1,..N\}$ and E the set of edges of G.

We define as *Music Transition Graph (MTG)* the digraph G with the additional following properties:

1. G is connected with no self loops and $|V|-1 \leq |E| \leq (|V|-1)^2 + |V| - 1$
2. every $e_{i,j} \in E$ represents a possible transition from vertex i to j during navigation with cost function $w_\tau(i,j)$
3. for every $i \in [0,N-1]$ there is at least one edge leaving vertex i $e_{i,i+1}$.
4. let d(i) the duration of a musical event $S_i \in \mathbf{S}$, $d_0 = 0$. The cost function of an edge $e_{i,j} \in E$ is defined as:
   - $w_\tau(i,j) = d(j)$, if $j = i+1$, $\forall i,j \in (0,N)$
   - $w_\tau(i,j) = 0$, if $j \neq i+1$, $\forall i,j \in [0,N]$
   - $w_\tau(0,1) = 0$.

Solution to problem

Let a path $p =< i_0, i_1, .., i_k >$ in a graph G' and the weight of path p the sum of the weights of its constituent edges:

$$w(p) = \sum_{j=1}^{k} w(i_{j-1}, i_j). \quad (2)$$

We define the shortest-path weight from s to t by:

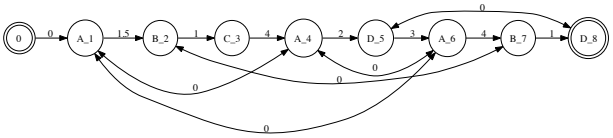


Figure 3. A Music Transition Graph for melody ABCADABD. Arc weights correspond to the note duration of the source. Bidirectional arcs above and below nodes connect patterns with common context.

$$\delta(s,t) = min\{w(p) : s \rightsquigarrow^p t\}. \quad (3)$$

Let a MTG G to stylistically represent the sequence S_n. This graph is connected, with no self loops and no negative weights. Given that, from definition 7, $w_\tau(i,j)$ is strictly by the duration $d(i,j)$ of graph events, our problem consists of finding a MTG's shortest path problem. The solution for the MTG shortest path exists and can be found in polynomial time. One of the most known solutions to this problem can be found with the help of Dijkstra's algorithm, with complexity $O(\log V)$. Dijkstra's algorithm does not solve the single pair shortest-path problem direcly, it solves instead the single-source shortest path problem, however no algorithms for the single pair shortest path problem are known that run asymptotically faster than the best single-source algorithms in the worst case.

Corollary from solution

Given a music sequence, the MTG representation permits solving the problem of accessing from a musical event s a musical event t within a music sequence S in the less possible time. Thus, by recombining musical events within S, we can reproduce a novel sequence from s to t. On the same time, this sequence respects the network of the MTG graph, hence the stylistic properties of the original music sequence.

Application 1

Suppose a music melody $S\{A,B,C,A,D,A,B,D\}$, with durations $d_S\{1.5,1,4,2,3,4,1,2.5\}$.

We construct a MTG $G(V,E)$ with edges $e(i,j) \in E$ with for $e(i,j) : j \neq i+1$ the arc connect common context according to the metric $m = PITCH$ (figure 3).

Suppose that our problem is to find the quickest possible transition from vertex $s = 2$ (note B) to vertex $t = 8$ (note D). To solve our problem, we can apply Dijkstra's algorithm.

We apply the algorithm for our sequence S and state D. When the algorithm terminates we have the shortest paths for all graph vertices. We can resume that for our problem the solution is the path $p =< B_2, B_7, D_8 >$. For the navigation along a path, we ignore one of two interconnected components. Hence, the final path is $p' =< B_2, D_8 >$.

We presented the formalisms and the solution to the simplest sequence scheduling problem. In our research, we focus on a number of problems that we are treating with the same methodology. These problems are combination of problems in sequence scheduling and sequence alignment domain. A list of the more important ones that we are dealing with is as follows:

1)Find the shortest path in time from a state s to a state t (examined).

2) The same with problem 1 with the constraint on the length of common context during recombinations.

3) Find the shortest path in time from a state s to a given sequence .

4) Find the continuation relatively to given sequence (Continuator).

5)The same with problem 1 with the additional constraint on the total number of recombinations.

6)Find a path from a state s to a state t with a given duration t, with $t_1 \leq t \leq t_2$.

7)Find a path from a state s to a state t with a given duration t, with $t_1 \leq t \leq t_2$ and with the additional constraint on the total number of recombinations (problem 5 + 6)

7. GRAIPE FOR COMPUTER ASSISTED IMPROVISATION

Our algorithms and architecture, are integrated in an under development software under the name GrAIPE. GrAIPE stands for Graph Assisted Interactive Performance Environment. It is an ensemble of modular objects for max-msp implementing the architecture presented in section 4.2. Concerning GrAIPE's design and implementation, our basic priorities are intelligent interfacing to the performer and efficient well-implemented algorithms for concurrency, interaction and the system's core main functions. Whether the software is under development, an instantiation of GrAIL has already taken place under the name PolyMax for machine improvisation, simulating with success 10 concurrent omax-like improvisers.

8. CONCLUSIONS - FUTURE RESEARCH

In this report we tried to set the basis for a novel, three-party interaction scheme and proposed a corresponding architecture for Computer Assisted Improvisation. Employing the approach of stylistic learning and stylistic interaction, we operated to formalize this interaction scheme under formalisms inspired from graph theory. We then discussed a simple music sequence scheduling problem.

Graph approach to CAI appears to be promising for modeling three-party interaction in a real-time non supervised improvisation environment that includes a 'silicon' participant. Not only does it permit a formalization of CAI problems in relation with space and time complexity, but it also approaches timing and capacity issues with widely accepted time-space units (for instance, milliseconds) that can make explicit the connection of our theoretical results with real-time system own formalism. This can be proved extremely practical for the future when integrating our theoretical results to real-time environment, in contrast with other formalisms such as in [18] that despite of penetrating complex interactivity issues, their temporal analysis in abstract time units makes this connection more implicit. Furthermore, graph formalization allows transversal bibliography research in all domains where graphs have been employed (production scheduling, routing and QoS etc.), and thus permit the generalization of music problems to universal problems and their confrontation with algorithms that have been studied in this vast both theoretic and applicative domain of graph theory.

In the recent future we are focusing on presenting formal solutions for the problems list in the previous section. Of particular interest are approximate solutions to problems 5, 6, 7, which are NP-hard for the general case.

Concerning development, GrAIPE has still a lot way to run until it fits with the requirements set in section 4.2. Even though already with a scheduler, a basic visualization module and a scripting module, these modules are being re-designed to adapt to the new research challenges. Other modules are a constraint-based user interface and its communication with a solver which is under development.

9. REFERENCES

[1] De Mantaras, R., Arcos, J., "AI and Music. From Composition to Expressive Performance" , AI Magazine, Vol. 23 No.3, 2002.

[2] Thom, B., "Articial Intelligence and Real-Time Interactive Improvisation", Proceedings from the AAAI-2000 Music and AI Workshop, AAAI Press, 2000.

[3] Assayag, G. , Bloch, G., "Navigating the Oracle: a Heuristic Approach", Proc. ICMC'07, The In. Comp. Music Association, Copenhagen 2007.

[4] Fry, C., "FLAVORS BAND: A language for Specifying Musical Style", Machine Models of Music, p. 427-451, Cambridge, MIT Press, 1993.

[5] Franklin, J. A. Multi-Phase Learning for Jazz Improvisation and Interaction. Paper presented at the Eighth Biennial Symposium on Art and Technology, 13 March, New London, Connecticut, 2001.

[6] Biles, A. GENJAM: A Genetic Algorithm for Generating Jazz Solos. In Proceedings of the 1994 International Computer Music Conference, 131137. San Francisco, Calif.: International Computer Music Association, J. A. 1994.

[7] Miranda, E., "Brain-Computer music interface for composition and performance", in International Journal on Disability and Human Development, 5(2):119-125, 2006.

[8] Graves, S.,"A Review of Production Scheduling", Operations Research, Vol. 29, No. 4, Operations Management (Jul. - Aug., 1981), pp. 646-675, 1981.

[9] Papadopoulos, G., and Wiggins, G. 1998. "A Genetic Algorithm for the Generation of Jazz Melodies". Paper presented at the Finnish Conference on Artificial Intelligence (SteP98), 79 September, Jyvaskyla, Finland, 1998.

[10] Giffler, B., Thompson, L., "Algorithms for Solving Production-Scheduling Problems", Operations Research, Vol. 8, No. 4, pp. 487-503, 1960.

[11] M., Cont, A., Dubnov, S.,Assayag, G., "Anticipatory Model of Musical Style Imitation Using Collaborative and Competitive Reinforcement Learning", Lecture Notes in Computer Science,2007.

[12] Pachet,F., "The continuator: Musical interaction with style". In proceedings of International Computer music Conference, Gotheborg (Sweden), ICMA,2002.

[13] Allauzen C., Crochemore M., Raffinot M., "Factor oracle: a new structure for pattern matching, Proceedings of SOFSEM'99, Theory and Practice of Informatics, J. Pavelka, G. Tel and M. Bartosek ed., Milovy, Lecture Notes in Computer Science 1725, pp 291-306, Berlin, 1999.

[14] Blackwell, T., Young, M. "Self-Organised Music". In Organised Sound 9(2): 123136, 2004.

[15] Collins, N., McLean, A., Rohrhuber, J., Ward, A., "Live coding in laptop performance", Organized Sound, 8:3:321-330, 2003.

[16] Dannenberg, R., "Real-TIme Scheduling And Computer Accompaniment Current Directions in Computer Music Research", Cambridge, MA: MIT Press, 1989.

[17] Puckette, M., "Combining Event and Signal Processing in the MAX Graphical Programming Environment", Computer Music Journal, Vol. 15, No. 3, pp. 68-77, MIT Press, 1991.

[18] Rueda, C., Valencia, F., "A Temporal concurrent constraint calculus as an audi processing framework", Sound and Music Computing Conference, 2005.

[19] http://www.audiosynth.com/

[20] http://chuck.cs.princeton.edu/

A PERCEPTUAL STUDY ON DYNAMICAL FORM IN MUSIC

Jens Hjortkjær
Dept. of Arts and Cultural Studies
University of Copenhagen, Denmark
jensh@hum.ku.dk

Frederik Nielbo
Center for Semiotics
University of Aarhus, Denmark
stud20074536@hum.au.dk

ABSTRACT

The concept of dynamical form is presented as a dimension of music perception. Dynamical form refers to the subjective perception of temporal events in music (*explosive, fading out, rising* etc.). In a behavioral experiment listeners were asked to categorize musical excerpts varying in musical period, tonality, instrumentation, and acoustic features while attending to their dynamical form. Data indicates that subjects are sensitive to dynamical forms, but were particularly sensitive to a specific one (*suspense*). We also discuss a method of categorizing dynamical forms in terms of force dynamics.

1. INTRODUCTION

In this paper, we introduce the concept of dynamical form as a fundamental dimension of musical experience. The concept of dynamical form is not related to 'dynamics' in the sense of loudness in music, but pertains to the ways in which musical change is subjectively perceived. Dynamical forms may be described in language by adjectives such as *rising*, *fading out*, *explosive*, *surging*, *lively*, *tentative*, *rushing* etc. Previous work relevant to dynamical form (such as [12],[15]) has typically examined it in relation to musical emotion, but we argue that it should be considered as a distinct dimension. For instance, a musical passage experienced as *angry* may be angry in an eruptive or slowly rising fashion.

Dynamical forms are general cross-modal patterns of experience grounded in bodily-kinesthetic experience. They have been studied in infant psychology ([7]) and suggested to be among the earliest and most fundamental forms of experience in humans. In this context they are referred to as *vitality contours* or *vitality affects*, and pertain to the way in which temporal events are felt and categorized. Some music theorists have suggested their relevance to musical experience (e.g., [18]), but experimental work on this is sparse (allthough [1] may be said to be related).

On the other hand, experimental work in psychoacoustics indicates that recognition of events or gestures (e.g., [4],[21],[28]), as well as physical properties of the sound source (e.g., [3],[5],[8],[22],[24]) are of fundamental relevance to auditory perception in general. Gestural aspects have also been suggested to be of relevance to music perception (see e.g. [2]). However, we consider gestures to be an instance of the more general category of dynamical form. A gesture may be experienced as having a certain dynamic form - e.g. eruptive or toning out - but dynamical forms may also be experienced in relation to physical events (an eruptive volcano or the fading out of a river flow).

2. TAXONOMY OF DYNAMICAL FORMS

With the above in mind, we can provisionally sketch a classification scheme of 3 general and independent dimensions of musical experience: (a) emotion or valence, (b) source properties, and (c) dynamical form. Each dimension gives rise to linguistic descriptions that may be used to illustrate them:

Emotion	**Source**	**Dynamical form**
Mourning	Crunchy	Fleeting
Angry	Heavy	Drawn out
Joyful	Compact	Explosive
Proud	Creaking	Floating
Gentle	Dirty	Expanding
Lamenting	Wet	Struggling

Table 1. Verbal descriptors of basic listening dimensions

The studies in music psychology most relevant to dynamical form are studies of musical 'tension' or 'force' ([10],[11],[23]). This may refer to either patterns of tension and relaxation in harmonic structure, or to the felt presence of tension due to rhythmic complexity, acoustic dissonance, melodic attraction, etc. A number of experimental studies have used a continuous response method to have subjects rate the amount of perceived tension in the course of music listening (e.g., [9],[11],[25]).

Unlike tension considered as a continuous variable, dynamical forms are semantic units of perceived events or states. While music may be experienced as a continuous flow of tension, we argue that dynamical forms arise from the experience of interaction between opposing tension or force tendencies. In physical experience, we constantly perceive interactions between objects having given force tendencies. We may perceive a man leaning against an unlocked door as an unstable scene where

there is a balance between the force applied by the weight of the man and the resisting force of the door. Perception of scenes like this possesses force dynamical structure in the sense described by L. Talmy ([14]) from the point of view of language. We will suggest that dynamical forms in music can also be categorized in terms of force dynamics.

2.1 A dynamical model of force semantics

Force dynamical patterns are characterized in terms of two interacting force entities with conflicting force tendencies. In the scene with the man leaning against a door there are two conflicting force tendencies. We perceive the man as potentially moving, but being withheld by the stronger force tendency (towards rest) of the door. Force dynamical patterns are constituted by the two opposing force tendencies (one towards action or motion, and one towards rest), and by the balance of strength between these tendencies. Perceiving (and describing) the scene also incorporates a certain perspective. For instance, one would tend to see the man leaning against the door from the perspective of the man (rather than viewing the door as being leaned-upon by the man).

Four basic patterns describing steady-state force oppositions arise from the possible interactions of these structural elements (force tendencies, balance of strength) and the perspective. Consider instead, for instance, a man that keeps standing in spite of a heavy wind blowing upon him. In this scene, the structural roles have switched: now the man has a tendency towards rest that is stronger than the opposing force tendency of the wind (towards action).

In addition to these steady-state patterns, corresponding changing-state patterns exist. If the door breaks and the man falls down, then there is a change in the structural elements: the weaker force tendency of the door is removed and the man's tendency towards motion is actualized. If the wind knocks the man off his feet, then the changing-state pattern is that of a shift in the balance of strength between the two force entities.

Such semantical force patterns have been described by L. Talmy within cognitive linguistics ([14]). They are considered to be fundamental cognitive forms underlying our understanding of basic phenomena such as causation and modality. In the present context, we will suggest that force dynamical patterns can be described at a general level as a dynamical system. Consider a one-dimensional dynamical system with 2 control parameters:

$$\dot{x} = h + rx - x^3 \qquad (1)$$

where h and r represent the magnitudes of the two interacting forces. A stable state of the system correspond to a given force tendency (towards action or rest). The system undergoes bifurcations when the two force parameters are varied continuously. Bifurcations occur around a cusp shape in the (r,h)-plane (see figure 1). Inside the cusp, there are two asymmetrically stable states (a conflict of forces), while there is only one outside. In Figure 1 below, stable states are illustrated by potential functions of the flow in given states (stable states corresponding to local minima). A 'ball' in the potential minimum indicates that the state is realized. The realized states are seen from the point of view of the first force entity.

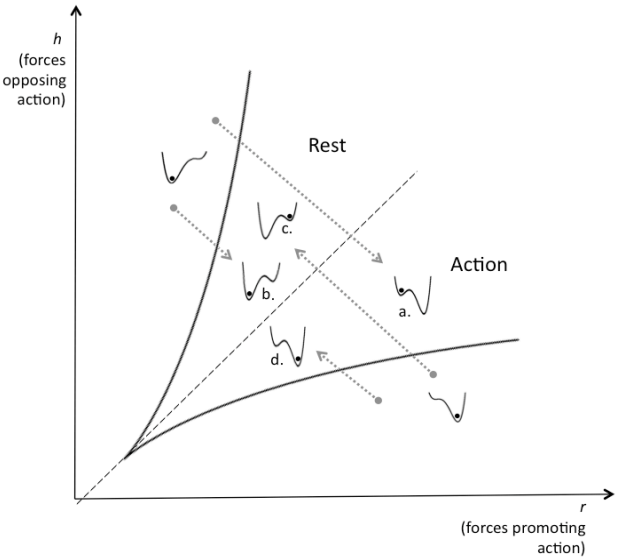


Figure 1. Stability diagram of the (r,h)-plane. Stable states are indicated by potential minima. Stippled arrows indicate the initial state of a given realized state.

The model follows the delay convention. This means that a stable state (a force tendency) persists until it is destroyed. Outside the cusp there is a single stable state corresponding to action or rest (corresponding to the dominating force). When entering the cusp, a conflict between two potential states (rest/action) arises. At the midline (h=r) there is a balance of strength between the forces, and trajectories across this line correspond to transitions between action and rest.

The force parameters (h,r) correspond to the perception of tension. Although these vary continuously there are only four qualitatively different conflicting states of the system (force dynamical steady state patterns *a*-*d*). There may be a tendency towards rest (*a*-*b*) or towards action (*c*-*d*). The alternative potential state is either more (*a*-*c*) or less (*b*-*d*) stable resulting in action (*a*-*d*) or rest (*b*-*c*). For instance, in state *c* we have the situation described above of the man leaning against a door. There is a tendency towards motion but a stronger tendency towards rest (causing rest). The man standing in spite of the wind appears as state *b* (a stronger tendency towards rest). We may readily extend the model to encompass changing-state patterns (where a transition between action and rest takes place). This corresponds to trajectories either across the cusp (e.g., caused or blocked actions), or across the midline (changed balance of strengths, e.g. trapped action, overcoming, etc.).

Dynamical forms can be categorized as generalized force dynamical patterns. This means that a dynamical form can be categorized into specified semantic types or dynamic 'archetypes'. The man leaning against the door (state *c*) corresponds to dynamical forms expressing *suspense* or *tension* (a state of suppressed action). Force magnitudes may vary continuously but the way in which they interact with other forces gives rise to a limited number of patterns. In relation to music we argue that continuously perceived tension gives rise to discrete semantic patterns (dynamical forms) that can be characterized by their force dynamical structure.

In Figure 2 below, dynamical forms of different steady-state types are shown in the bifurcation diagram described by verbal labels:

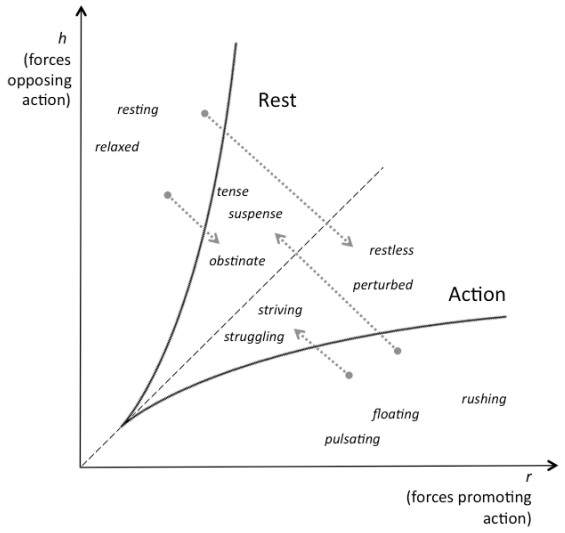


Figure 2. Verbal labels indicating dynamical forms in the stability diagram

3. EXPERIMENT

3.1 Experimental design

A behavioral experiment was designed to examine whether listeners are sensitive to dynamical forms in music. Subjects performed two different listening tasks on 16 short musical excerpts of existing performed music. The excerpts where chosen by the authors to represent a broad variation on a number of musical parameters: tonality, mode, instrumentation, genre, and period of production. The excerpts could all be considered 'classical music', but included both tonal and atonal music, older and more recent music, as well as different instrumentations. Excerpts of 10-15 seconds were taken from the following works:

No	Composer	Work	Tonality	Instr.	Year
1	Liszt	Paganini etude no. 2 (La Campanella)	tonal (mi)	piano	1851
2	Mahler	Symphony no. 3, 6th movement	tonal (ma)	orch.	1896
3	Messiaen	Vingt regards sur l'enfant jesus, no. X	atonal	piano	1944
4	Prokofiev	Romeo & Juliet, finale	atonal	orch.	1935
5	Alva Noto	Attack (UTP_)	atonal	orch.	2009
6	Beethoven	Piano Sonata, op. 31, 1st movement	tonal (mi)	piano	1803
7	Debussy	Preludes for piano, no. 3 (Animé)	tonal (ma)	piano	1910
8	Grisey	Modulations	atonal	orch.	1978
9	Abrahamsen	Boogie Woogie	atonal	piano	1983
10	Bartok	String quartet no. 4, 4th movement	atonal	orch.	1929
11	Brahms	Cappricio, op. 116, no. 2	tonal (mi)	piano	1878
12	Tchaikovsky	Slavonic march, op. 31	tonal (mi)	orch.	1852
13	Barry	The sinking of Devonshire (Soundtrack)	tonal (mi)	orch.	1997
14	Ligeti	Atmosphères	atonal	orch.	1961
15	Schubert	Piano sonata, D784, 1st movement	tonal (mi)	piano	1815
16	Smetana	On the seashore	tonal (mi)	piano	1861

Table 2. List of excerpts used in the experiment

In choosing the musical excerpts, the authors found 4 different dynamic forms to be most salient. These may be described as (i) *overcoming, climaxing* (no. 1-4), (ii) *eruptive*, *bursting* (no. 5-8), (iii) *striving, struggling* (no. 9-12), and (iv) *suspense, tense* (no. 13-16). These correspond to four different force dynamical patterns, as discussed above. Forms i-ii are changing-state patterns in which there is either an overcoming (i) or removal (ii) of an antagonistic force. Forms iii-iv correspond to steady-state patterns c and d respectively in figure 1 (also indicated in figure 2).

However, subjects were not informed of those forms. They were simply instructed to attend to dynamical form in general. The concept was introduced to subjects beforehand by two different examples of dynamical forms in other excerpts. The question of interest was whether subjects would group the excerpts according to perceived dynamical form across major musical categories (tonality, period, instrumentation).

Fourteen subjects of varying musical background participated in the experiment. The experiment consisted of two separate parts. In the first part of the experiment, similarity ratings were obtained. Subjects were presented with triads of excerpts. In each trial, subjects were asked to indicate which two excerpts of the three were most similar, and to rate their similarity on a discrete rating scale (1-7). Each subject was not presented with all possible combinations of excerpts, but the between-subject order of presentation was fixed so that the subjects collectively heard all combinations. The second part of the experiment consisted of a free clustering task. All 16 excerpts could be accessed, and subjects where asked to group each of them into four groups or less. They were also asked to write word labels indicating how they perceived a given excerpt. The subjects were allowed to write as many labels they liked. They gave between 1-3 labels for each excerpt.

3.2 Results

A mean similarity matrix between each excerpt was created for each task. For the first task, similarities simply corresponded to ratings. In the free clustering task, the similarities were calculated as percentage overlap between groups ([19]). The matrices produced by the different tasks were highly correlated, even though the subjects were presented with different subsets of excerpt combinations in the first task. Consequently, the two matrices are collapsed in the following analysis.

A hierarchical cluster analysis (single-linkage) was performed on the mean similarity data. Four groups are seen in the analysis, indicated in figure 3. These groups were - interestingly - not identical to the ones hypothesized by the authors. For each group, the word labels given in the free clustering task were semantically highly related across subjects. Labels representative of these semantically related categories are shown next to the groups in figure 3.

The found group structure was compared with a number of factors that could be suspected to influence the perception of similarity between excerpts. A correlational analysis was performed, and only instrumentation was found to correlate significantly with the behavioral data. Neither period of production nor tonality or mode correlated significantly with the data. A number of psychoacoustic parameters of potential perceptual relevance were obtained by computerized analysis of the audio data. This included energy RMS, roughness ([27]), spectral irregularity ([13]), Mel-frequency cepstrum coefficients, tempo, flatness of the temporal envelope, and slope of the spectral centroid. All of these were found to have low correlation (< .24) with the behavioral data.

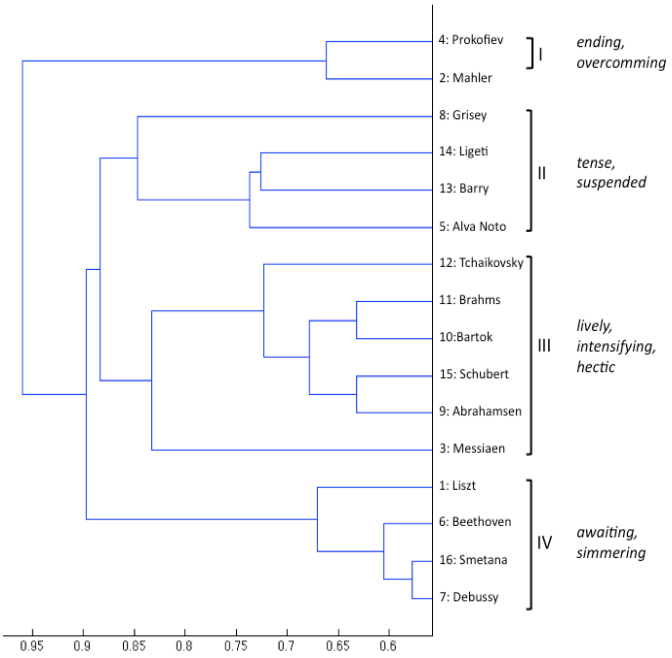


Figure 3. Hierarchical cluster analysis of behavioral data with verbal labels given by subjects

3.3 Discussion

The results suggest that listeners when instructed to attend to dynamical forms, group musical excerpts across musical periods, genres, tonality, and acoustic features. The separation in terms of instrumentation into two broad groups (piano/orchestra) does not account for all variation in the similarity data. This could suggest that the subjects are in fact sensitive to dynamical forms that are independent of these factors. The verbal labels given by subjects also indicated their sensitivity to dynamical forms.

Interestingly, the similarities perceived by subjects were different from the ones suspected by the authors. A given musical excerpt may possess a number of dynamical forms simultaneously. For instance, excerpts 5 and 8 contain sudden differences in instrumentation and loudness that may be experienced as *explosive* or *eruptive*. But they also have an undefined harmonic structure that may cause them to be experienced as *tense* or expressing a state of *suspense* at a more global level. Apparently, this second quality appeared more salient than the first.

The moderate clarity of the behavioral data and the resulting clustering may be caused by the fact that subjects attended to dynamical forms that are of similar type. Three of the four groups (II-IV) have identical force dynamical structure. They are different instances of force dynamical pattern *c*: a force tendency towards motion with a stronger opposing force. Subjects thus preferably attend to dynamical forms expressing a state of tension or suspense. Only group I has different force dynamical structure (a changing event pattern), expressing a transition from motion to rest (*ending*, *overcoming*).

It is interesting to observe the apparent saliency of pattern *c* in music. Theories of musical listening are often based on the idea that music is subjectively perceived as

motion ([16],[17],[20],[26]). The salient dynamical form found in this experiment, however, is not perceived as motion, but as an interaction of forces resulting in rest as suspended motion. In this way we may argue that force dynamical structure represents a more fundamental way of conceiving the subjective nature of music perception.

The similarity of perceived dynamical forms makes it difficult draw conclusions on the basis of these behavioral results. We would suggest that additional experiments with other musical excerpts, perhaps reflecting an even broader range of musical variation, is needed to gain more information about the perception of dynamical forms.

4. REFERENCES

[1] A. Frey, C. Marie, L. Prod'homme, M. Timsit-Berthier, D. Schön, M. Besson: "Temporal semiotic units as minimal meaningful units in music? An electrophyiological approach," *Music Perception*, vol. 26, iss. 3, pp. 247-256, 2009

[2] A. Gritten, E. King, (Eds.): *Music and gesture*, Ashgate, Hampshire, 2006

[3] A.J. Kunkler-Peck, M.T. Turvey: "Hearing shapes," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 26, no. 1, pp. 279-294, 2000

[4] B.H. Repp, "The sounds of two hands clapping: An exploratory study," *Journal of the Acoustical Society of America*, vol. 81, iss. 4, pp. 1100-1109, 1987

[5] C. Carello, K.A. Anderson, A.J. Kunkler-Peck: "Perception of object length by sound," *Psychological Science*, vol. 9, pp. 211-214, 1998

[6] C. Krumhansl: "A perceptual analysis of Mozart's piano sonata K. 282: Segmentation, tension, and musical ideas," *Music Perception*, vol. 13, no. 3, pp. 401-432, 1996

[7] D. Stern: "Vitality contours: The temporal contour of feelings as a basic unit for constructing the infant's social experience," In P. Rochat (Ed.), *Early social cognition. Understanding others in the first months of life*, Lawrence Erlbaum, London, pp. 67-80, 1999

[8] F. Avancini, D. Rocchesso: "Controlling material properties in physical models of sounding objects," *Proceedings of the International Computer Music Conference 2001*, pp. 91-94, 2001

[9] F. Lerdahl, C. Krumhansl: "Modeling tonal tension," *Music Perception*, vol. 24, iss. 4, 329-366, 2007

[10] F. Lerdahl: *Tonal Pitch Space*. Oxford University Press, New York, 2001

[11] F.V. Nielsen: *Oplevelsen af musikalsk spænding* [The experience of musical tension]. Akademisk Forlag, Copenhagen, 1983

[12] K. Hevner: "Experimental studies of the elements of expression in music," *The American Journal of Psychology*, vol. 48, no. 2, pp. 246-268, 1936

[13] K. Jensen: *Timbre models of musical sounds*. DIKU, Copenhagen, 1999

[14] L. Talmy: "Force dynamics in language and cognition," In L. Talmy: *Toward a Cognitive Semantics*. MIT, Cambridge, pp. 409-470, 2000

[15] M. Clynes, N. Nettheim: "The living quality of music: Neurobiological patterns of communicating feeling," In M. Clynes (Ed.), *Music, mind, and brain: The neuropsychology of music*, Plenum Press, New York, 1982

[16] M.R. Jones: "Music as stimulus for psychological motion: Part I. Some determinants of expectancies," *Psychomusicology*, vol. 1, pp. 34-51, 1981

[17] M.R. Jones: "Music as stimulus for psychological motion: Part II. An expectancy model," *Psychomusicology*, vol. 2, pp. 1-13, 1982

[18] O. Kühl: *Musical Semantics*, Peter Lang, Bern, 2007

[19] P. Dunn-Rankin, G.A. Knezek, S. Wallace, S. Zhang. *Scaling methods*, 2nd ed., Lawrence Erlbaum Associates, London, 2004

[20] R. Gjerdingen: "Apparent motion in music," *Music Perception*, vol. 11, no. 4, pp. 335-370, 1994

[21] R.E. Pastore, J.D. Flint, J.R. Gaston, M.J. Solomon: "Auditory event perception: The source-perception loop for posture in human gait," *Perception & Psychophysics*, 2008, vol. 70, iss. 1, pp. 13-29, 2008

[22] S. Lakatos, S. McAdams, R. Caussé: "The representation of auditory source characteristics: Simple geometric form," *Perception & Psychophysics*, vol. 59, no. 8, pp. 1180-1190, 1997

[23] S. Larson: "Musical forces and melodic expectations: Comparing computer models and experimental results," *Music Perception*, vol. 21, no. 4, pp. 457-498, 2004

[24] S. McAdams, A. Chaigne, V. Roussarie, "The psychomechanics of simulated sound sources: Material properties of impacted bars," *Journal of the Acoustical Society of America*, vol. 115, iss. 3, pp. 1306-1320, 2004

[25] S. McAdams, B.W. Vines, S. Vieillard, B.K. Smith, R. Reynolds: "Influences of large-scale form on continuous ratings in response to a contemporary piece in a live concert setting," *Music Perception*, vol. 22, no. 2, pp. 297-350, 2004

[26] V. Zuckerkandl: *Sound and symbol: Music and the external world*. Pantheon Books, New York, 1956

[27] W. Sethares: *Tuning, Timbre, Spectrum, Scale*. Springer, New York, 1998

[28] X.F. Li, R.L Logan, R.E. Pastore, "Perception of acoustic source characteristics: Walking sounds,"

Structural Modeling Of Pinna-Related Transfer Functions

Simone Spagnol spagnols@dei.unipd.it
Michele Geronazzo geronazz@dei.unipd.it
Federico Avanzini avanzini@dei.unipd.it
Università di Padova

ABSTRACT

This paper faces the general problem of modeling pinna-related transfer functions (PRTFs) for 3-D sound rendering. Following a structural *modus operandi*, we exploit an algorithm for the decomposition of PRTFs into ear resonances and frequency notches due to reflections over pinna cavities in order to deliver a method to extract the frequencies of the most important spectral notches. Ray-tracing analysis reveals a convincing correspondence between extracted frequencies and pinna cavities of a bunch of subjects. We then propose a model for PRTF synthesis which allows to control separately the evolution of resonances and spectral notches through the design of two distinct filter blocks. The resulting model is suitable for future integration into a structural head-related transfer function model, and for parametrization over anthropometrical measurements of a wide range of subjects.

1. INTRODUCTION

Back in 1907, Lord Rayleigh's Duplex Theory of Localization [1] gave birth to the vast and still little understood field of 3-D sound. As a matter of fact, the well-known diffraction formula which approximates the behaviour of a sound wave produced by an infinite point source around the listener's head provided a first glimpse of what we today call a head-related transfer function (HRTF). Nevertheless, compared to such centenary theory, most of the relevant issues in HRTF modeling are relatively recent.

There exists a number of ways to render HRTF-based spatial audio. Approximations based on low-order rational functions [2] and series expansions of HRTFs [3] were proposed, resulting in simple yet valuable tools for HRTF modeling. On the other hand the complexity of filter coefficients and weights, respectively, makes both techniques unsuitable for real-time applications. Conversely, structural modeling [4] seems nowadays to be an attractive alternative approach for all those scenarios where fast computation is needed: within this characterization, the contribution of the listener's head, ears and torso to the HRTF are isolated in several subcomponents, each accounting for some well-defined physical phenomenon. Thanks to linearity, the global HRTF can be later reconstructed from a proper combination of all these effects. The result of such a decomposition is a model which is both economical and well-suited to real-time implementations. Furthermore, the intuitive nature of physical parameters gives us the possibility of relating the model to anthropometrical measurements.

Our work focuses on the contribution of the pinna to the HRTF. Although perceptually dominated by head motion cues, pinna effects on incident sound waves are of great importance in sound spatialization. Several experiments have shown that, contrarily to azimuth effects which are dominated by diffraction around the listener's head and may be reduced to simple and intuitive binaural quantities, elevation cues are basically monaural and heavily depend on the listener's pinna shape, being the result of a superposition of scattering waves influenced by a number of resonant modes inside pinna cavities. Within this framework, it is crucial to find a suitable model for representing the pinna contribution to the HRTF, whose transfer function we commonly refer to as Pinna-Related Transfer Function (PRTF). In addition, linking the model parameters to straightforward anthropometric measurements on the user's pinnas is equally relevant and represents the ultimate challenge in this direction. Once such model is available, cascading it to a simple Head-and-Torso (HAT) model [5] would yield a complete structural HRTF representation.

In this paper we carry on the work presented in [6]. Exploiting an iterative approach which aims at separating resonance effects from pinna reflections in experimentally measured PRTFs, a method for extracting the frequencies of the most important notches is here sketched, followed by a discussion on the possible relation between notch frequencies and anthropometry. Finally, a structural model of the pinna is proposed.

2. PREVIOUS WORKS

Following Batteau's studies [7], high-frequency components which arrive at the listener's ear are typically reflected by the concha wall and rim, provided that their wavelength is small compared to the pinna dimensions. Due to interference between the direct and reflected waves, sharp notches can be observed in the incoming sound's spectrum at high frequencies with a periodicity of $1/\tau_i$, where τ_i is the time delay of the i-th reflection. Such observation led to a very simple double-path model of the pinna [8], whose main drawback lies in fixed reflection coefficients that overestimate the effective number of notches in the spectrum. Even so, the model's fit with experimental data was found to be reasonably good.

A similar approach was adopted by Barreto *et al.*, whose

model [9] consists in a reflection structure represented by four parallel paths, each modeled by a time delay τ_i and a magnitude factor ρ_i, cascaded to a low-order resonator block. As a matter of fact pinna cavities act as resonators, affecting the frequency content of both the direct and the reflected sound waves (see Shaw's work [10] for an extensive analysis of pinna resonant modes). The model parameters are fitted by decomposing each specific measured head-related impulse response (HRIR) into four scaled and delayed damped sinusoidal components using a procedure based on the second-order Steiglitz-McBride (STMCB) algorithm, and associating the delay and scaling factor of each component to the corresponding parameters of its associated path in the model. Multiple regression analysis was used in order to link the model parameters to eight measured anthropometric features [11]. Unfortunately, besides having no clear evidence of the physics behind the scattering phenomenon, the considered measures can only be obtained with the help of a 3-D laser scanner. Regardless of such particular concerns, this work surely certifies our final PRTF model's architecture, viz. a "resonance-plus-delay" structure.

The approach taken by Raykar *et al.* for reflection modeling [12] is different and operates both in the time and frequency domains. The authors used robust digital signal processing techniques based on the residual of a linear prediction model for the head-related impulse response (HRIR) to extract the frequencies of the spectral notches due to the pinna alone. Specifically, first the autocorrelation function of the HRIR's windowed LP residual is computed; then, frequencies of the spectral notches are found as the local minima of the group-delay function of the windowed autocorrelation. In addition, a ray-tracing argument was exploited to attest that the so found spectral notches are somehow related to the shape and anthropometry of the pinna. For each of the extracted notches the corresponding distance was plotted on the image of the pinna, and by varying elevation such mapping appeared consistent with reflections on the back of the concha and on the crus helias. Spectral peaks were extracted in parallel by means of a linear prediction analysis, yielding results which match quite well the pinna resonant modes reported by Shaw [10] and further justifying the "resonance-plus-delay" approach.

Hitherto, another significant contribution on low-cost modeling of PRTFs was provided by Satarzadeh *et al.* [13]. In this work PRTFs for elevation $\phi = 0^\circ$ are synthesized through a model composed of bandpass and comb filters, which respectively approximate the two major resonances (Shaw's resonant modes 1 and 4) and one main reflection. The two second-order bandpass filters and the comb filter are interconnected as in Figure 1, the latter taking the form $[1 + \rho \exp(-sT)]$, where T is the time delay of the considered reflection estimated from the spacing of notches in the PRTF spectrum and ρ a frequency-dependent reflection coefficient which strongly attenuates low-frequency notches, coming over Batteau's model aforementioned limitation. For what concerns the resonant part, a cylindrical approximation to the concha is used, where depth and width of the cylinder uniquely define the depth resonance, while

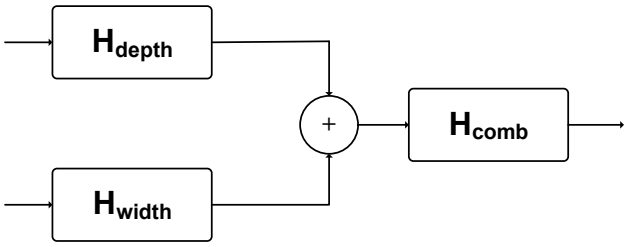


Figure 1. *The PRTF model proposed by Satarzadeh* et al. *[13].*

the width resonance is thought to be correlated to the time delay T depending on whether the concha or the rim is the significant reflector. Though the anthropometric significance of the two parameters is not robust, Satarzadeh claimed that if the pinna has an approximately cylindrical shaped concha and a structure with a dominant reflection area (concha or rim), his anthropometry-based filter provides a good fit to the experimental PRTF. In particular, it features sufficient adaptability to fit both PRTFs with rich and poor notch structures.

3. PRTF ANALYSIS

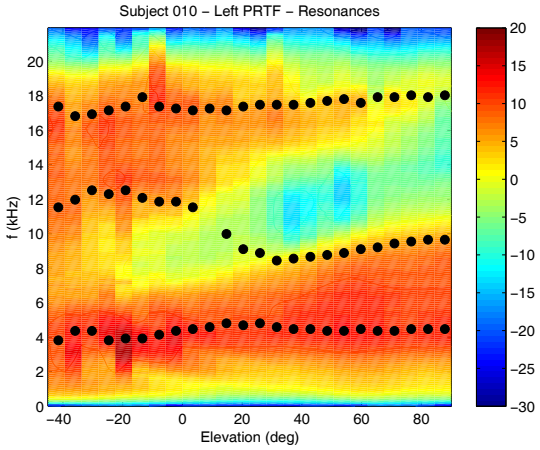
Taking the last two works described in the previous section as an inspiration and a "resonance-plus-delay" PRTF model as starting point, the main and final goal of our work is the construction of an essential multi-notch filter suitable for anthropometric parametrization. This obviously requires a PRTF analysis step. In order to analyze PRTFs, we consider measured HRIRs from the CIPIC database [14], a public domain database of high spatial resolution HRIR measurements at 1250 directions for 45 different subjects along with their anthropometry. We choose to investigate the behaviour of pinna features in subjects 010, 027, 134, and 165 in order to facilitate comparison with previous works on notch frequencies extraction (the same subjects' PRTFs were analyzed in [12]).

3.1 Pre-processing

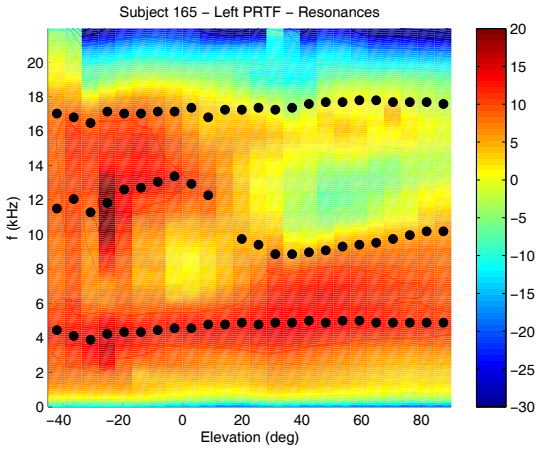
For purpose of analysis we focus on HRIRs sampled on the median plane, with elevation varying from -45° to 90°. As a matter of fact, since sensitivity of PRTFs to azimuth is weak [13], we roughly expect PRTFs to be elevation dependent only. Such an assumption makes the PRTF model suitable for all azimuths.

Knowing that the magnitude response of an earless head with respect to a sound source in the median plane is ideally flat if the head is modeled as a rigid sphere, the only preprocessing step we apply to obtain a raw estimate of the PRTF is windowing the corresponding HRIR using a 1.0 ms Hann window [12]. In this way, spectral effects due to reflections caused by shoulders and torso are removed from the PRTF estimate.

In order to isolate the spectral notches in the so built PRTFs we exploit an ad-hoc designed algorithm that returns an estimate of the separated resonant and reflective components. The idea behind such algorithm is to itera-

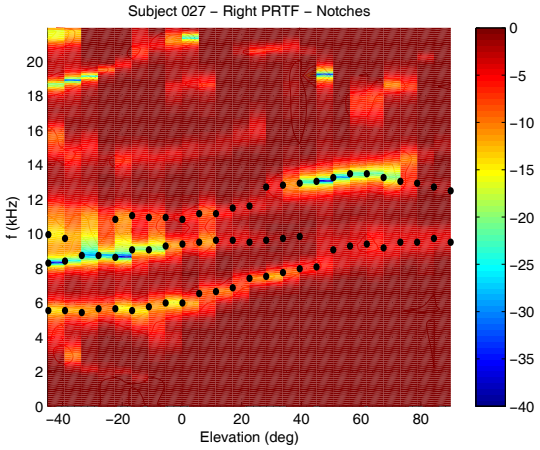


(a) Subject 010.

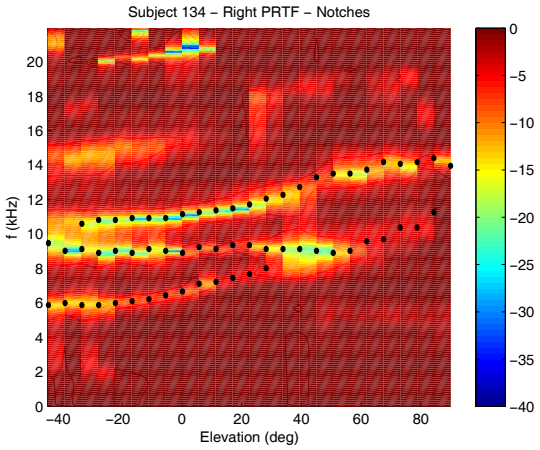


(b) Subject 165.

Figure 2. *Resonance plots for different elevations.*



(a) Subject 027.



(b) Subject 134.

Figure 3. *Spectral notch plots for different elevations.*

tively compensate the PRTF magnitude spectrum with an approximate multi-notch filter until no significant notches are left. Once convergence is reached, say at iteration n, the PRTF spectrum embodies the resonant component alone, while a combination of the n multi-notch filters provides the reflective component. A detailed and accurate description of the separation algorithm is reported in [6].

3.2 Resonances

We now discuss the PRTF features identified by the decomposition carried out through the separation algorithm. From the 3-D plots in Figure 2 we can study how the resonances' contribution for Subjects 010 and 165 varies throughout all available elevations. The center frequency of each resonance was extracted with the help of an identification system based on a sixth-order ARMA model [15] and spatially tracked along elevation, resulting in the dotted tracks superposed on the plots.

We can easily identify two major hot-colored areas in these plots. The first one, centered around 4 kHz, appears to be very similar amongst subjects since it spans all elevations. One may immediately notice that this area includes Shaw's omnidirectional mode 1. The resonance's bandwidth appears to increase with elevation; however, knowledge of pinna modes implies that a second resonance is likely to interfere within this frequency range, specifically Shaw's mode 2 (centered around 7 kHz with a magnitude of 10 dB). On the other hand, the second hot-colored area differs both in shape and shade amongst subjects. Still it is most prominent at low elevations between 12 and 18 kHz, a frequency range which is in general agreement with Shaw's horizontal modes 4, 5, and 6.

Note that the higher resonance may be perceptually irrelevant since it lies near the upper limit of the audible range. In addition, since the resonances at 12 and 7 kHz are excited in mutually exclusive elevation ranges, we may look forward to a double-resonance filter design.

3.3 Notches

Similarly to the resonance plots, those in Figure 3 represent the frequency notches' contribution for Subjects 027 and 134. As expected, reflection patterns strongly depend on elevation and pinna shape. While PRTFs generally exhibit poor notch structures when the source is above the head, as soon as the elevation angle decreases the number and depth of frequency notches grows to an extent that varies

among subjects.

However, several analogies can be noticed here too. In order to investigate such common trends, we inherit an analysis tool that is widely used in the field of sinusoidal modeling, specifically the McAulay-Quatieri partial tracking algorithm (see [16] for details), to track the most prominent notches' patterns along all elevations. Originally, this algorithm was used to group sinusoidal partials (extracted through a peak detection algorithm) along consecutive temporal windows according to their spectral location. We implemented the original version [16] of the algorithm; obviously, since in our case elevation dependency replaces temporal evolution and spectral notches take the role of partials, we call it "notch tracking" algorithm. The notch detection step simply locates all of the local minima in the reflective component's spectrum, while the matching interval for the notch tracking procedure is set to $\Delta = 3$ kHz.

Since it is preferable to restrict our attention to the frequency range where reflections due to the pinna alone are most likely seen, and ignore notches which are overall feeble, two post-processing steps are performed on the obtained tracks:

- delete the tracks which are born and die outside the range $4 - 14$ kHz;
- delete the tracks that do not present a notch deeper than 5 dB.

The outputs of the notch tracking algorithm are the dotted tracks superposed on the plots in Figure 3. Results are definitely akin to those found in [12] with the use of an elaborated DSP-based algorithm. Three major tracks are seen for both subjects, whereas the shorter track in Subject 027's plot very probably represents the continuation of the missing track at those elevations. Reasonably, the gap between tracks is caused by the algorithm's impossibility of locating proper minima in that region (due e.g. to superposition of two different notches or the presence in the magnitude plot of valleys which are not notch-like). However, the three longer tracks suggest that similar reflection patterns occur in different PRTFs.

3.4 Reflections and anthropometry

We now move towards the definition of a realistic mapping between notch frequencies and reflection points over the pinna, by relating each major notch at frequency f_0 to a different reflection (assuming it to be the first and most prominent notch of a periodic series). Reflection models typically assume that all reflection coefficients are positive. In such case, in order for destructive interference to occur (viz. for notches to be produced in the spectrum), the extra distance travelled by the reflected wave with respect to the direct wave must be equal to half a wavelength:

$$t_d = \frac{1}{2f_0}. \tag{1}$$

This assumption was used in [12] to trace reflection points over pinna images based on the extracted notch frequencies.

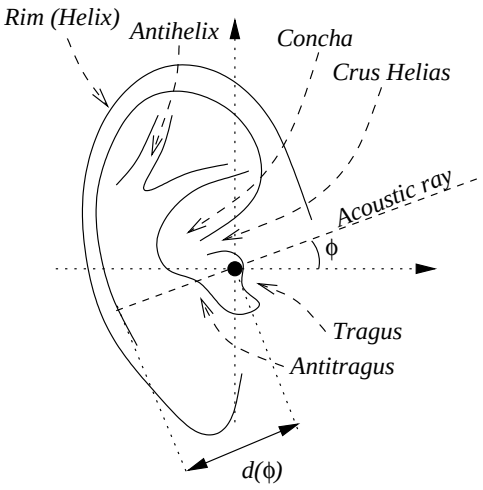


Figure 4. *Anatomy of the pinna.*

However, in [17] it was pointed out that over 80% out of a test bed of 20 CIPIC subjects exhibit a clear negative reflection in the HRIR. With the help of a simple physical model of the pinna the authors argued that, since the impedance of the pinna is greater than that of air, there may be a boundary created by an impedance discontinuity which could produce its own reflection and ultimately reverse the phase of the wave. In this latter case, a delay of half a wavelength would not produce notches in the spectrum any more. Instead, destructive interference would appear for full-wavelength delays only:

$$t_d = \frac{1}{f_0}. \tag{2}$$

We choose to use this last assumption, and relate notches to pinna geometry through a simple ray-tracing procedure similar to the one described in [12].

The distance of each reflection point with respect to the entrance of the ear canal is calculated through the following equation,

$$d(\phi) = \frac{ct_d(\phi)}{2} = \frac{c}{2f_0(\phi)}, \tag{3}$$

where $f_0(\phi)$ represents the frequency of the current notch at elevation ϕ and c is the speed of sound (approximately 343 m/s). The assumption of negative reflection coefficient causes distances to be roughly doubled with respect to those computed in [12]. Then, considering the 2-D polar coordinate system illustrated in Figure 4 with the right ear canal entrance as origin, each notch is mapped to the point $(d(\phi), \pi + \phi)$.

Results for Subject 134 are reported in Figure 5. The so-obtained mapping reveals a high degree of correspondence between calculated reflection points and pinna geometry:

- the track nearest to the ear canal very closely follows the concha wall, with a slight displacement at low elevations probably caused by the little extra distance needed by the wave to pass over the crus helias;
- the intermediate track can be associated to a reflection on the rim's edge and on the antihelix;

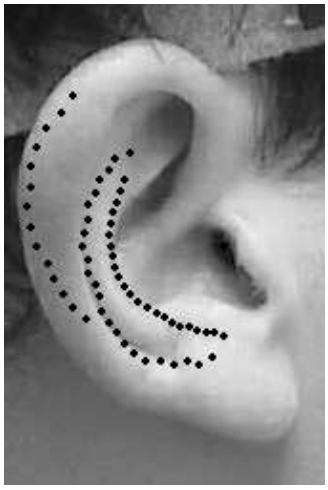


Figure 5. *Reflection points on Subject* 134*'s right pinna.*

- the furthest track follows the shape of the rim and stops in the vicinity of the point where the rim terminates, hence it is likely to be associated to a reflection in the inner wall of it.

This analysis, that yields convincing results for other subjects too, opens the door for a very attractive approach to the parametrization of the structural PRTF model based on individual anthropometry. Given a 2-D image (or possibly a 3-D reconstruction) of the user's pinna, it is possible to trace the contours of the concha wall, antihelix and rim, compute their distances with respect to the ear canal entrance, and derive the notch frequencies by reversing Eq. (3). Obviously, in order to fully justify these findings, robust theoretical motivations and a rigorous analysis using a vast test bed of subjects are required. Furthermore, since notch depth varies strongly with subjects and elevations, the reflection coefficient must also be estimated for each point.

4. A STRUCTURAL MODEL OF THE PINNA

The information gathered from the outputs of the decomposition and notch tracking algorithms allows to model the PRTF with two resonances and three spectral notches. As Figure 6 depicts, our final aim is to design two distinct filter blocks, one accounting for resonances and one for reflections. Clearly, in order to reach complete control of the filter parameters, full parametrization of the model on anthropometrical measurements is needed. Hence for the moment we shall present the PRTF re-synthesis procedure driven by the outputs of the two above algorithms.

4.1 Filter design

In Section 3.2 we have shown that a PRTF at one specific elevation includes two main resonances in the frequency range of interest for the pinna. It is then possible to approximate the effective resonances by deducing center frequency f_C and magnitude G of each resonance from the dotted tracks and directly using the so found parameters to design two second-order peak filters with fixed bandwidth $f_B = 5$ kHz of the form [18]

$$H_{res}(z) = \frac{V_0(1-h)(1-z^{-2})}{1+2dhz^{-1}+(2h-1)z^{-2}}, \tag{4}$$

where

$$h = \frac{1}{1+\tan(\pi\frac{f_B}{f_s})}, \tag{5}$$

$$d = -\cos(2\pi\frac{f_C}{f_s}), \tag{6}$$

$$V_0 = 10^{\frac{G}{20}}, \tag{7}$$

and f_s is the sampling frequency. A posteriori analysis of the synthesized resonances has revealed that PRTFs for high elevations only need the first resonance to be synthesized, being the second very close to it. We thus choose to bypass the second resonant filter when $\phi \geq 20°$.

Similarly, for what concerns the reflection block, we feed the center frequency f_C, notch depth G, and bandwidth f_B parameters coming from the notch tracking algorithm to three second-order notch filters of the form [19]

$$H_{refl}(z) = \frac{1+(1+k)\frac{H_0}{2}+d(1-k)z^{-1}+(-k-(1+k)\frac{H_0}{2})z^{-2}}{1+d(1-k)z^{-1}-kz^{-2}}, \tag{8}$$

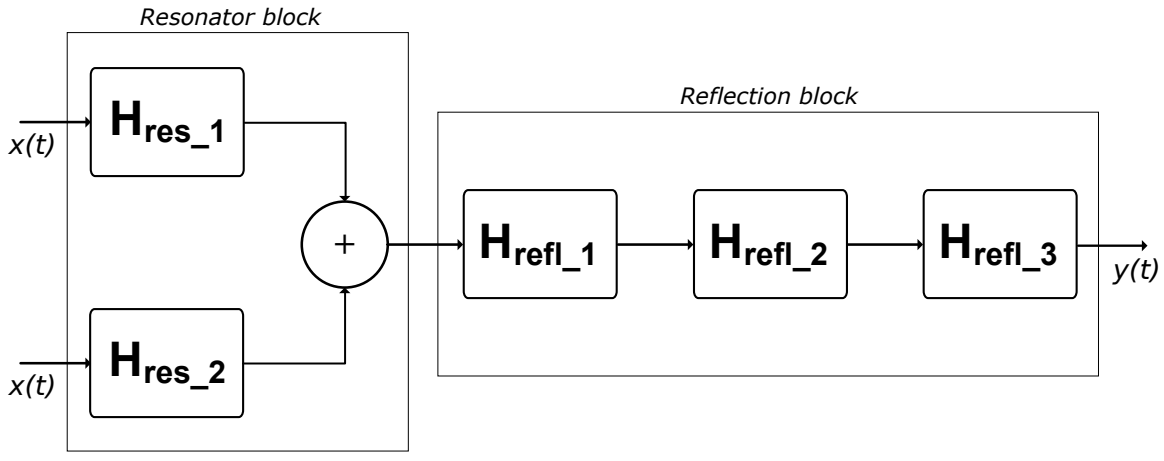


Figure 6. *General model for the reconstruction of PRTFs.*

where d is defined as in Eq. (6) and

$$V_0 = 10^{-\frac{G}{20}}, \tag{9}$$

$$H_0 = V_0 - 1, \tag{10}$$

$$k = \frac{\tan(\pi \frac{f_B}{f_s}) - V_0}{\tan(\pi \frac{f_B}{f_s}) + V_0}, \tag{11}$$

each accounting for a different spectral notch. The three notch filters must be placed in series and cascaded to the parallel of the two peak filters, resulting in an eighth-order global filter.

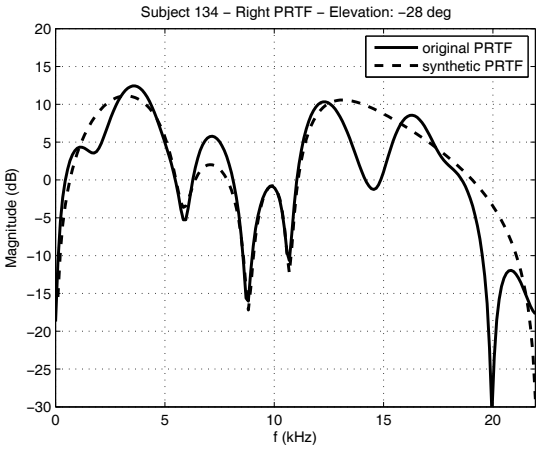
4.2 Results

Figure 7 reports the comparison between original and re-synthesized PRTF magnitudes for three distinct subjects, each at a different elevation. Adherence rate to the original PRTFs is overall satisfactory in the frequency range up to 14 kHz. Still, several types of imperfections need to be adjusted: as a first example, deep frequency notches that appear at low elevations complicate the notch filter design procedure. In point of fact, if the notch to be approximated is particularly deep and sharp, the second-order filter will produce a shallower and broader notch whose bandwidth may interfere with adjacent notches, resulting in underestimating the PRTF magnitude response in the frequency interval between them. Figures 7(a) and 7(b) show this behaviour around 7.5 and 10 kHz, respectively. Using a filter design procedure which forces to respect the notch bandwidth specification during re-synthesis would grant a better rendering of resonances, at the expense of worsening notch depth accuracy.

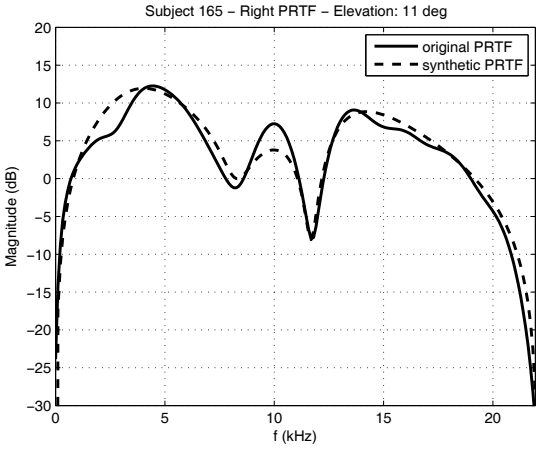
The absence of modeled notches over the upper frequency threshold is another cause of imprecision. For instance, Figure 7(a) presents an evident mismatch between original and modeled PRTF just after the 12.5-kHz peak, due to the cut of the frequency notch at 14.5 kHz. This problem may be corrected by increasing the 14-kHz threshold in order to take into account a higher number of notches. However, being the psychoacoustic relevance of this frequency range relatively low, the effective weight of the mismatch is reduced.

Last but not least, resonance modeling may bring approximation errors too. In particular, the possible presence of non-modeled interfering resonances and the fixed-bandwidth specification both represent a limitation to the re-synthesis procedure. Furthermore, center frequencies extracted by the ARMA identification method mentioned in Section 3.2 do not always coincide with peaks in the PRTF. Thus a stronger criterion for extracting the main parameters of each resonance is needed. Nevertheless, the approximation error seems to be negligible in all those cases where resonances are distinctly identifiable in the PRTF.

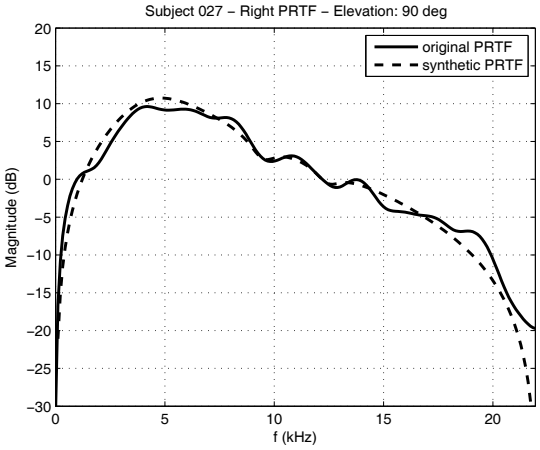
In conclusion, the above presented re-synthesis model appears to be overall effective, especially for PRTFs which clearly show one or two main resonant modes and moderately deep notches. Figure 7(c) supports this assertion.



(a) Subject 134, elevation $-28°$.



(b) Subject 165, elevation $11°$.



(c) Subject 027, elevation $90°$.

Figure 7. *Original vs Synthetic PRTF plots.*

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach for structural PRTF modeling, which exploits an algorithm that separates the resonant and reflective parts of the PRTF spectrum. We used such decomposition to re-synthesize the original PRTF through a low-order filter model, whose results show an overall suitable approximation. In a parallel manner, our attempt towards the explanation of the scattering process resulting in the most important spectral notches in the PRTF provided visually convincing results. Besides improving the synthesis step, ongoing and future work includes understanding of the reflection coefficient and relating the resonant component of the PRTF to anthropometry.

6. REFERENCES

[1] J. W. Strutt, "On our perception of sound direction," *Philosophical Magazine*, vol. 13, pp. 214–232, 1907.

[2] E. C. Durant and G. H. Wakefield, "Efficient model fitting using a genetic algorithm: pole-zero approximations of HRTFs," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 1, pp. 18–27, 2002.

[3] D. J. Kistler and F. L. Wightman, "A model of head-related transfer functions based on principal components analysis and minimum-phase reconstruction," *J. Acoust. Soc. Am.*, vol. 91, no. 3, pp. 1637–1647, 1992.

[4] C. P. Brown and R. O. Duda, "A structural model for binaural sound synthesis," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 5, pp. 476–488, 1998.

[5] V. R. Algazi, R. O. Duda, and D. M. Thompson, "The use of head-and-torso models for improved spatial sound synthesis," in *Proc. 113th Convention of the Audio Engineering Society*, (Los Angeles, CA, USA), 2002.

[6] M. Geronazzo, S. Spagnol, and F. Avanzini, "Estimation and modeling of pinna-related transfer functions," in *Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*, (Graz, Austria), September 6-10 2010. Accepted for publication.

[7] D. W. Batteau, "The role of the pinna in human localization," *Proc. R. Soc. London. Series B, Biological Sciences*, vol. 168, pp. 158–180, August 1967.

[8] A. J. Watkins, "Psychoacoustical aspects of synthesized vertical locale cues," *J. Acoust. Soc. Am.*, vol. 63, pp. 1152–1165, April 1978.

[9] K. J. Faller II, A. Barreto, N. Gupta, and N. Rishe, "Time and frequency decomposition of head-related impulse responses for the development of customizable spatial audio models," *WSEAS Transactions on Signal Processing*, vol. 2, no. 11, pp. 1465–1472, 2006.

[10] E. A. G. Shaw, *Binaural and Spatial Hearing in Real and Virtual Environments*, ch. Acoustical features of human ear, pp. 25–47. Mahwah, NJ, USA: R. H. Gilkey and T. R. Anderson, Lawrence Erlbaum Associates, 1997.

[11] N. Gupta, A. Barreto, and M. Choudhury, "Modeling head-related transfer functions based on pinna anthropometry," in *Proc. of the Second International Latin American and Caribbean Conference for Engineering and Technology (LACCEI)*, (Miami, FL, USA), 2004.

[12] V. C. Raykar, R. Duraiswami, and B. Yegnanarayana, "Extracting the frequencies of the pinna spectral notches in measured head related impulse responses," *J. Acoust. Soc. Am.*, vol. 118, pp. 364–374, July 2005.

[13] P. Satarzadeh, R. V. Algazi, and R. O. Duda, "Physical and filter pinna models based on anthropometry," in *Proc. 122nd Convention of the Audio Engineering Society*, (Vienna, Austria), May 5-8 2007.

[14] R. V. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The CIPIC HRTF database," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, (New Paltz, New York, USA), pp. 1–4, 2001.

[15] P. A. A. Esquef, M. Karjalainen, and V. Välimäki, "Frequency-zooming ARMA modeling for analysis of noisy string instrument tones," *EURASIP Journal on Applied Signal Processing: Special Issue on Digital Audio for Multimedia Communications*, no. 10, pp. 953–967, 2003.

[16] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[17] P. Satarzadeh, "A study of physical and circuit models of the human pinnae," Master's thesis, University of California Davis, 2006.

[18] S. J. Orfanidis, ed., *Introduction To Signal Processing*. Prentice Hall, 1996.

[19] U. Zölzer, ed., *Digital Audio Effects*. New York, NY, USA: J. Wiley & Sons, 2002.

CONNECTING GRAPHICAL SCORES TO SOUND SYNTHESIS IN PWGL

Mika Kuuskankare
Centre for Music and Technology
Sibelius Academy
Helsinki, Finland
mkuuskan@siba.fi

Mikael Laurson
Department of Signal Processing and Acoustics
Aalto University
Espoo, Finland
laurson@siba.fi

ABSTRACT

In this paper we describe how graphical scores can be coupled with synthesis algorithms in the visual programming language PWGL. The present approach is based on an extensible music notation and a direct connection to a flexible sound synthesis engine. We implement, as an exercise, a simple working model that makes it possible create graphical scores out of user defined graphical objects and connect the graphical objects to specific synthesis methods.

1. INTRODUCTION

Creating a computer-based, extensible music notation system that can be tied directly to sound synthesis is a long-standing problem in the field of computer music. The integration between the two major tools, ENP [1] and PWGLSynth [2], inside the visual programming environment PWGL [3], is a step to this direction. There exists a long tradition of tools that explore the possibilities of combining graphics to sound synthesis. Lindemann's Animal [4] and Buxton's SSSP [5] were among the first experiments. A more up-to-date graphical score editor implementation can be found inside Pd as described by Puckette [6]. The UPIC [7] system developed by Iannis Xenakis in the mid 70's is perhaps the most notable example in this category and there have been several attempts, such as the open source IanniX [8], to recreate it using commodity hardware. However, most of these attempts concentrate solely on graphical notation.

The approach presented here makes no difference between a traditionally notated score and a graphical one. For example, the two ENP scores shown in Figure 1 both share the same underlying representation and they can both be created, manipulated, and performed in the same way. Up to now, our attempts of realizing performance models for ENP have been concentrated on controlling virtual instruments and more traditional sound sources, such as MIDI and sample libraries.

In this paper, we aim to demonstrate that it is relatively easy the extend PWGL so that it makes it possible also to realize graphical scores. We show how to implement a PWGL playback device that can be used to play, through PWGLSynth, graphical scores prepared with the help of ENP. We describe the protocol that can be used to create new graphical symbols, demonstrate how to define the specific synthesis algorithms and how to create the connection between the two.

The rest of the paper is structured as follows. First, we give a concise introduction to the playback scheme of PWGL and build a minimal playback device. Next, we discuss how to define visual synthesis instruments and collections thereof, i.e., orchestras. Then, we show how to create customized graphical objects and demonstrate how to control the aspects of visualization and synthesis through the use of properties. Finally, we put it all together by creating the connection between our synthesis instruments and the graphical objects. The paper ends with some concluding remarks.

2. THE IMPLEMENTATION

Figure 6 in the Appendix shows our target score lasting for about 20 seconds containing some user designed graphical objects. The score can be edited with the mouse and objects can be added, deleted, copied, and pasted using keyboard shortcuts. The vertical positioning, size and the duration of the objects can be edited by mouse drags. Other aspects can be controlled by using user-definable properties. The benefits and efficiency of using properties has already been reported by Dannenberg (see, for example [9]). Properties are name/value pairs and they can be manipulated either manually (through the ENP GUI) or algorithmically. In our case, the resulting sound can depend on or be defined by the graphical attributes. Conversely, the appearance of the graphical objects can be generated according to the synthesis properties.

The example score contains two kinds of objects: (1) a streaming sound sample player object that displays the sample data contained in a file named by the *:filename* property, and (2) a sine-bank object that reads its frequency information from a sound analysis database and interpolates between two sets of frequencies. The frequency information retrieved from the analysis is stored by the synthesis algorithm as a property. The data which, in turn, is used when drawing the object.

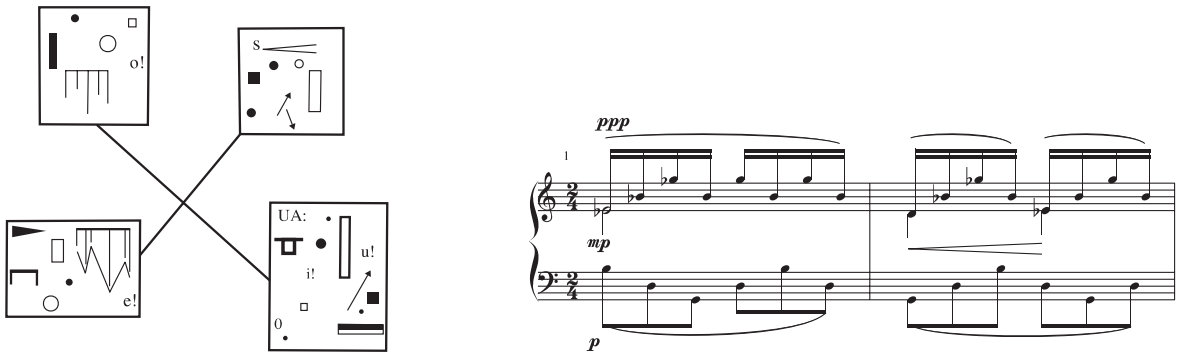


Figure 1. (a) a graphical score, and (b) a traditionally notated score both prepared with the help of ENP. These scores share the same underlying object representation and hierarchy.

2.1 The Playback Device

PWGL Playback Devices are used to generate performance data (e.g, MIDI) for playback of ENP scores. A new playback device can be created by subclassing the abstract *pwgl-playback-device* class and specializing the pertinent methods. The performance data, in turn, is generated by calling a set of methods in a pre-defined order. Figure 2 shows the primary methods and their order of execution. As a general rule, the *prepare-playback* is used to initialize the player, e.g., by loading a sample database or preparing a specific instrument setup. *add-playback-note-event* method inserts each note found in the score into an event queue. *setup-playback* method, in turn, can be used to send arbitrary messages, such as, volume or pitch-bend information before the playback starts. Finally, during the realtime playback, *send-playback-event* is called for each event found in the event queue.

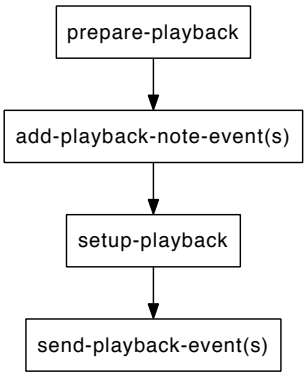


Figure 2. The pertinent PWGL playback methods and their order of execution.

Listing 1 shows the implementation of the PWGL playback device that can be used to play graphical scores.

We begin to implement our playback device by subclassing the *pwgl-onset-playback-device* class (a) which is specialized for sound sources that transmit only one playback event per note (as opposed to pwgl-onset-offset-play-device which implements a control protocol, that applies among others to MIDI, where the onset of a note and the offset of the same note are considered as two separate events). In our case, this is sufficient, as the duration of an event can be controlled, for example, with an amplitude envelope.

Listing 1 The definition of the PWGL playback device.

```
(a) (defclass GS-player (pwgl-onset-play-device) ())

(b) (defmethod prepare-playback* ((self GS-player) score)
(c)   (register-GS-instruments)
(d)   (compile-GS-expressions score)
(e)   (start/stop-GS-player))

(f) (defmethod send-playback-event ((output GS-player)
(g)                                 midi-info &optional vel?)
(h)   (dolist (expression (expressions note))
(i)           (funcall (read-key expression :rt-event-fn))))
```

Next, *prepare-playback** (b-e) registers all the available synthesis instruments, generates the playback code for the score objects (see *compile-GS-expressions* in line d) and caches the information for efficiency (see Section 2.3). Also, depending on the state of the synthesizer, it either starts or stops the playback process.

Finally, we need to specify how the actual playback event is sent to the output device (PWGLSynth in this case). The synthesis events have been compiled as function calls by the *compile-GS-expressions* method. Thus, *send-playback-event* (f-i) simply executes the cached functions one at the time (see line i).

2.2 Defining the Graphical Objects

Next, we create our custom ENP-expressions[10] and define their visual appearance using the built-in expression editor, the ENP Expression Designer (ED, [11]). For our experimental player we need two kinds of graphical objects: one that represents sound samples, and another that represents a sine bank with two sets of sine waves.

Figure 3 gives, as an example, the complete ED session that is used to define the sound sample expression. As our system is object-oriented we define a new ENP-expression class through inheritance. In this case the class name, *sample*, is given at the top left part of the editor. This class identity is of primary importance as it is later used to tie

together the graphical object created here and the methods that generate the actual synthesis information.

The visual representation is defined at the upper part of the editor using the combination of Lisp and a set of pre-defined OpenGL macros. Here, we first draw a rectangle (*draw-2D-box*) whose y dimension is chosen arbitrarily and the x dimension corresponds to the duration of the sample. In addition, we draw the value of the *:filename* property above the rectangle, and finally, a built-in function *draw-sound-object* is used to draw the sample data inside the enclosing rectangle. A preview of the expression is displayed at the lower part of the editor.

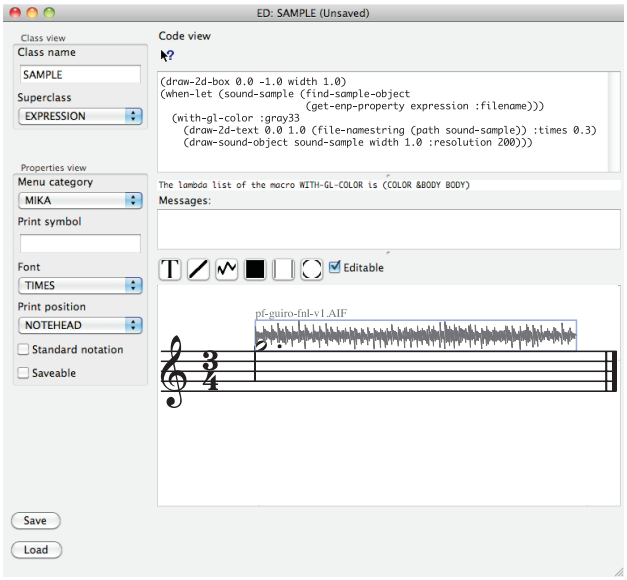


Figure 3. ENP expression designer can be used to define the objects needed for realizing a graphical score. Here, an object visually representing a sound sample is realized.

To save some space, for our second object we give only the code that defines its appearance (see Listing 2). Here, we assume that the two properties *:freqs1* and *:freqs2* each contain a list of frequencies. The two lists of frequencies are pre-calculated by the part of our scheme that generates the corresponding synthesis algorithm. The meaning of these lists is explained in Section 2.3. At this point, it suffices to know that the graphical output should reflect that fact that the synthesis algorithm interpolates between the two frequency sets so that the frequencies defined by *:freqs1* gradually fade out and, conversely, the frequencies defined by *:freqs2* fade in. The code is again relatively straightforward. Lines a-b access the frequency properties. Lines c-d retrieve the upper and lower bounds of the whole frequency envelope. The remaining parts of the code consist of two almost identical section, lines f-i and j-m, that draw the appropriate frequency sets using colors that fade in and out accordingly. Finally, in (n) we enclose the object inside a rectangle.

As can be seen it is relatively easy to define a new graphical objects in ENP. The process is interactive and the results are immediately usable. The graphical output

Listing 2 The drawing code of the sine-bank object.

```
(a) (let ((freqs1 (get-enp-property expression :freqs1))
(b)       (freqs2 (get-enp-property expression :freqs2)))
(c)   (let ((maxf (max (apply #'max freqs1) (apply #'max freqs2)))
(d)         (minf (min (apply #'min freqs1) (apply #'min freqs2))))
(e)     (with-GL-line-width 0.5
(f)       (with-2D-object :lines
(g)           (dolist (freq (pw::g-scaling freqs1 -1.0 1.0 minf maxf))
(h)             (with-GL-color :white (add-2d-vertex 0.0 freq))
(i)             (with-GL-color :black (add-2d-vertex width freq))))
(j)       (with-2D-object :lines
(k)           (dolist (freq (pw::g-scaling freqs2 -1.0 1.0 minf maxf))
(l)             (with-GL-color :black (add-2d-vertex 0.0 freq))
(m)             (with-GL-color :white (add-2d-vertex width freq)))))))
(n) (with-GL-color :gray (draw-2d-box 0.0 -1.1 width 1.1))
```

can be changed at any time and the changes are dynamically propagated across all the instances of the object in question.

2.3 Defining the Synthesis Algorithms

Figure 4 shows the visual instrument definition of our sample player. The connection between the visual instrument definition shown here and the specific synthesis algorithm is based on the boxes named *synth-plug*. These boxes define control entry points for our instrument. The ones labelled "D" (as in discrete) allow us to update values. The synth-plug boxes labelled "T", in turn, are used as triggers. All synth-plug boxes have as their first input the name of the control parameter (e.g. :amp, :trig, etc.). These symbolic references allow the user to refer to specific inputs while sending control events to the instrument.

In our visual instrument definition, the lines a, c, and e are of most importance (cf. Section 2.4). The *:set-env* and *:trig-env* (a) are used to store the incoming envelope data and trigger it respectively. The two entry points in (c) are used to control the built-in sample-player box (d). *:id* is used to select the sample from the internal sample database and *:trig* is used to trigger the sample player to start playing.

Furthermore, since we are talking about a multi-timbral player, we also need to define a patch that acts as our orchestra definition. As shown in Figure 5 the orchestra patch contains both instruments (*sine-bank* and *sampler*) and connects them to a PWGLSynth player box. The abstraction box containing the sound sample player, the implementation of which is shown in Figure 4, is labeled "Sampler", and can be seen at the top right corner of the patch.

2.4 Connecting the Graphics to Sound Synthesis

So far, we have defined a collection of new ENP-expressions and synthesis instruments. Next, we need to create the connection between the two. To accomplish our task we take an advantage of a newly created *RT-events* syntax [12] which can be used to express control information using Lisp forms. This syntax allows us to write Lisp expressions which, in turn, can be used to control low-level synthesis parameters of our synthesizer, e.g., to trigger events and write control values to specific inputs. The connection between the graphical objects and the visual instrument definition is implemented so that for every

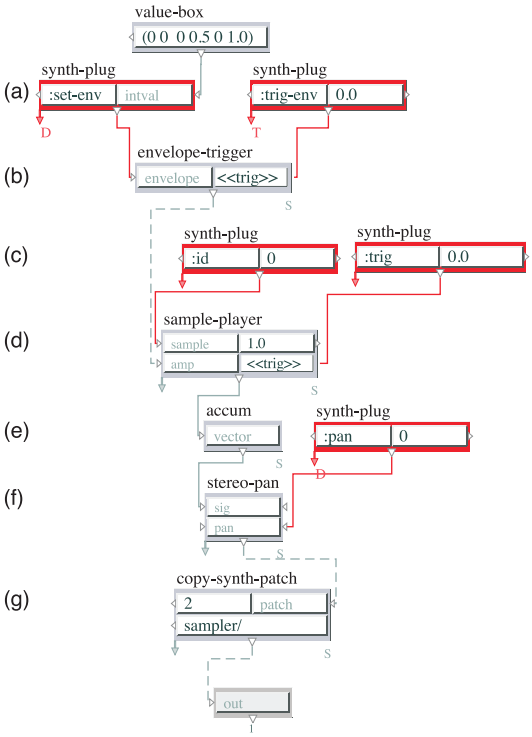


Figure 4. A PWGL patch showing the implementation of a simple sample player.

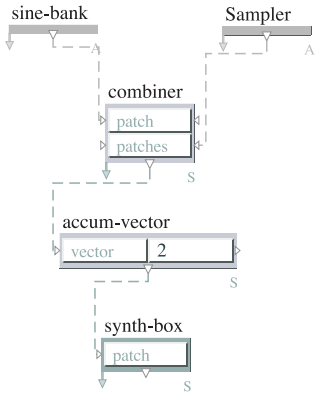


Figure 5. The "synth orchestra" patch containing two synth instruments *sine-bank* and *sampler*.

ENP-expression we provide a method called *expression-2-RT-events* which generates control information in the RT-events format. Furthermore, these methods can read from and write to the associated graphical objects any freely chosen property. Thus, the information generated here can be used to control some visual aspects. Also, the properties defined by the user or by the graphical object (such as duration or pitch) can, in turn, be used as a variable in the control data calculation.

Let us next examine the *expression-2-RT-events* method written for the sample player object in more detail. As explained, the functionality of this object depends on the *:filename* property. We used it in our visualization code to access the correct sample data for drawing. We also needed the property in our visual instrument definition as an argument to one of the synth-plug boxes (*:id*).

In Listing 3 we connect the graphical object to the visual instrument definition shown in Figure 4 by implementing the corresponding expression-2-RT-events method. The *:filename* property is accessed in our code in (b). The next lines do various checks (e.g., if the sample exists) and initializations, such as loading the sample by demand and accessing the numeric sample id required by the sample-player. Note, that due to the deliberate use Lisp backquote syntax, the samples are loaded when the playback data is generated and cached and not during realtime playback. In lines c-d the combination of the synth-event *:set-env* and the synth-trigger *:trig-env* is used to store and make active the amplitude envelope information using the appropriate entry points found in our visual instrument definition (see (a) in Figure 4). Finally, (e) – (f) are used to set the appropriate sample id (*:id*) and to trigger the sample-player box to render the sound (*:trig*). Thus, the implementation is now complete.

Listing 3 The RT-events code implementing the discrete and continuous control events used to drive our visual instrument defined in Figure 4.

```
(a) (defmethod expression-2-RT-events ((self sample))
(b)   (when-let (filename (get-enp-property self :filename))
(c)     (let ((sample (sample-object-exists-p filename)))
(d)       (unless sample
(e)         (setq sample (load-sound-sample (namestring filename))))
(f)       (let* ((dur (duration sample))
(g)              (id (sampleID sample)))
(h)         `(with-synth-instrument ,(GS-player-instrument :sampler) 1
(i)            (synth-event 0.0 :set-env ',(make-envelope '((0.0 0.3)
(j)                                                         (0.98 0.3)
(k)                                                         (1.0 0.0))
(l)                                                        "amp env"
(m)                                                        dur)
(n)                          :id 1 :namespace :sampler)
(o)            (synth-trigger 0.0 :trig-env :id 1 :namespace :sampler)
(p)            (synth-event 0.0 :id ,id :id 1 :namespace :sampler)
(q)            (synth-trigger 0.0 :trig :id 1 :namespace :sampler))))))
```

3. DISCUSSION

Using the concepts presented here it would be relatively straightforward to implement, for example, a simple but working multi-channel sample player à la ProTools. As an extension we could display the amplitude envelope or panning as an overlay to the sample objects. Several ENP-expressions are already able to display break-point functions as a part of the musical texture. Therefore, it would

be relatively easy to implement this feature as we could incorporate the envelope functionality, including display and editability, through inheritance.

4. CONCLUSIONS

In this paper we describe how graphical scores can be coupled with arbitrary synthesis algorithms in the visual programming language PWGL. We demonstrate how to implement a PWGL playback device that can be used to play a graphical score. We use the built-in editor to create graphical objects that can represent arbitrary complex synthesis algorithms (e.g., sound samples and resonator banks). We also demonstrate how to use user definable properties to control aspects of both sound and graphical objects. The main contribution of this paper is to show how an extensible music notation system can be connected to a flexible sound synthesis engine.

5. ACKNOWLEDGMENTS

The work of Mika Kuuskankare and Mikael Laurson has been supported by the Academy of Finland (SA 114116 and SA 122815).

6. REFERENCES

[1] M. Kuuskankare and M. Laurson, "Expressive Notation Package," *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[2] M. Laurson, V. Norilo, and M. Kuuskankare, "PWGLSynth: A Visual Synthesis Language for Virtual Instrument Design and Control," *Computer Music Journal*, vol. 29, pp. 29–41, Fall 2005.

[3] M. Laurson, M. Kuuskankare, and V. Norilo, "An Overview of PWGL, a Visual Programming Environment for Music," *Computer Music Journal*, vol. 33, no. 1, pp. 19–31, 2009.

[4] E. Lindemann and M. de Cecco, "Animal–a rapid prototyping environment for computer music systems," *Computer Music Journal*, vol. 15, no. 3, pp. 78–100, 1991.

[5] W. Buxton, R. Sniderman, W. Reeves, S. Patel, and R. Baecker, "The evolution of the sssp score editing tools," in *Foundations of Computer Music* (C. Roads and J. Strawn, eds.), pp. 376–402, MIT Press, Cambridge MA, 1985.

[6] M. Puckette, "Using Pd as a score language," in *Proceedings of International Computer Music Conference*, pp. 184–187, 2002.

[7] H. Lohner, "The UPIC system: A user's report," *Computer Music Journal*, vol. 10, pp. 42–49, Winter 1986.

[8] "Iannix." http://sourceforge.net/projects/iannix/

[9] R. B. Dannenberg, "A Structure for Efficient Update, Incremental Redisplay and Undo in Graphical Editors," *Software-Practice and Experience*, vol. 20, no. 2, pp. 109–132, 1990.

[10] M. Kuuskankare and M. Laurson, "ENP-Expressions, Score-BPF as a Case Study," in *Proceedings of International Computer Music Conference*, (Singapore), pp. 103–106, 2003.

[11] M. Kuuskankare and M. Laurson, "ENP Expression Designer - a Visual Tool for Creating User Definable Expressions," in *International Computer Music Conference*, (Barcelona, Spain), pp. 307–310, 2005.

[12] M. Laurson, V. Norilo, and M. Kuuskankare, "Compositional sketches in PWGLSynth," in *Proceedings of Dafx'09 Conference*, (Como, Italy), September 2009.

A. APPENDIX

Figure 6. A graphical score realized with the help of ENP. The objects shown here are user definable and can depend on or be defined by editable properties or synthesis algorithms connected to them.

CROWDSOURCING A REAL-WORLD ON-LINE QUERY BY HUMMING SYSTEM

Arefin Huq
Northwestern University
EECS Department
2133 Sheridan Road
Evanston, IL 60208, USA
fig@arefin.net

Mark Cartwright
Northwestern University
EECS Department
2133 Sheridan Road
Evanston, IL 60208, USA
mcartwright@u.northwestern.edu

Bryan Pardo
Northwestern University
EECS Department
2133 Sheridan Road
Evanston, IL 60208, USA
pardo@northwestern.edu

ABSTRACT

Systems able to find a song based on a sung, hummed, or whistled melody are called Query-By-Humming (QBH) systems. Tunebot is an online QBH web service and iPhone app that connects users to the desired recording on Amazon.com or iTunes. Tunebot's searchable database is composed of thousands of user-contributed melodies. Melodies are collected from user queries, sung contributions and through contributions from on-line play of an associated iPhone Karaoke game: Karaoke Callout. In this paper we describe the architecture and workings of the paired systems, as well as issues involved in building a real-world, working music search engine from user-contributed data.

INTRODUCTION

Music audio is one of the most popular categories of multimedia content. Examples include the song repositories of Apple's popular iTunes (www.apple.com/itunes), the indie-music site CD Baby (www.cdbaby.com) and Amazon (amazon.com). These music collections are indexed by such metadata as title, composer, and performer. Finding the desired recording with this indexing scheme can be a problem for those who do not know the metadata for the desired piece.

If the user has access to a recording of the desired audio (e.g. it is currently playing on the radio), then an audio fingerprinting system, such as Musiwave [1] or Shazam [2] can be used. Such systems require the query example be a (possibly degraded) copy of the exact recording desired. This makes audio fingerprinting unsuitable for any situation where the user is unable to provide a portion of the exact recording sought (e.g. the song ended on the radio before a search could begin).

Another approach is to identify a song based on entering its lyrics into a standard text-based search engine. This is a relatively mature field with successful commercial search engines (e.g. Google) already available. It is not, however, applicable to pieces of music that have no lyrics, or in situations where the user remembers the melody but not the words.

In this work, we concentrate on the situation where the user queries a system by singing or humming some portion of the song ("What is the name of the song that goes like this?"). Song identification systems that take sung or hummed input are known as query-by-humming (QBH) systems [3-4]. These are an example of melodic search engines. Melodic search engines (including QBH and rhythmic search) have received much attention in recent years [5-14] and use a melodic fragment as a query, entered as musical notation, through a virtual piano keyboard or sung into a microphone.

Most published research in QBH has focused on the matching algorithms and distance measures for melodies. While this is important, there are other technical and scientific challenges that must be surmounted to build an effective QBH system ready for real-world deployment. Example issues include: creation of a large database of relevant search keys, handling large numbers of users, speeding search as the database goes from hundreds to hundreds of thousands of melodies, and updating the database and matching algorithms after deployment in a seamless way.

Our solutions to these problems are embodied in *Tunebot*, an online QBH web service that connects users to the desired recording on Amazon.com or iTunes. Tunebot's searchable database is composed of thousands of user-contributed melodies. Melodies are collected from user queries, sung contributions and through contributions from on-line play of an associated Karaoke game: *Karaoke Callout*. In this paper we describe the architecture and workings of the paired systems, as well as issues involved in building a real-world, working music search engine from user-contributed data.

TUNEBOT

We embody our solutions to the problems of real-world QBH in an on-line web service called Tunebot (tunebot.org). Tunebot lets the user search for music by singing a bit of it (with or without lyrics) as a query.

The system does not require hand-coded search keys, since it automatically updates the database with new search keys derived from user queries and contribu-

tions. To speed data collection and encourage collaborative participation from the public, we integrate Tunebot with an online social music game (Karaoke Callout) that encourages collaborative tagging of audio with new search keys [15]. Karaoke Callout is a Game With A Purpose [16] that helps build our knowledge base of songs. Users may also register with our website and freely contribute sung examples in a manner similar to the OpenMind initiative [17].

User Interaction

Tunebot is available as a web service and is currently in beta testing as an iPhone application. The user interaction in both the web and iPhone versions is identical: 1) Sing, 2) Choose. The user simply sings a portion of the desired song to Tunebot. The system returns a ranked list of songs. Each song is playable by a simple click. While the song is playing, the system presents a dialog box asking if this is the correct piece of music. If the user clicks "yes," the query is stored in our database as a searchable example for that song. The user is then connected to either Amazon.com or iTunes where the music may be purchased. Figure 1 illustrates this interaction on the iPhone version of Tunebot. Figure 2 illustrates the Flash-based web interface for Tunebot.

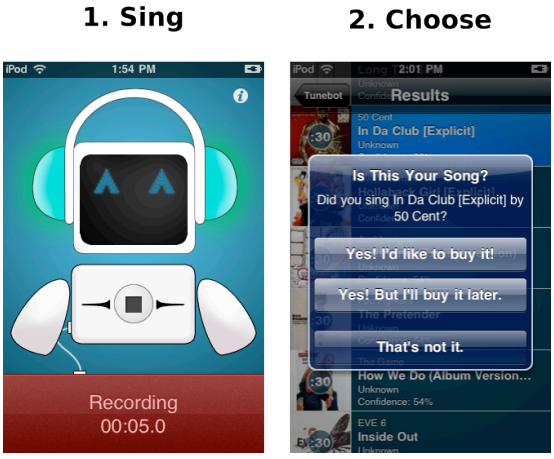


Figure 1. Screen shots of the iPhone interface for Tunebot.

Searchable Database Construction

Creating searchable keys that can be queried by singing is non-trivial. Hand-keying a database with thousands or millions of documents consumes prohibitive amounts of effort, as does updating and vetting such a database as music is created and tastes change. Thus, it is important to develop good methods to create and vet perceptually relevant search keys for music audio that allow the creation of a large music database indexed by musical content. This database must be expandable after deployment so the system may search for new music introduced as time goes by. For a system to scale, this must be done with minimal training and minimal oversight by human operators.

We do not currently use existing MIDI files or extraction of melodies from the original polyphonic audio. Automated transcription of polyphonic commercial recordings is still not sufficiently robust to provide good searchable melodies. The symbolically encoded databases available to us do not provide the coverage of modern pop, and rock tunes that our users tend to search for. Further, as user tastes change and new songs are released, a real working system must have the ability to constantly add songs to the database after deployment. We address these issues by turning to the users of the system for contributions.

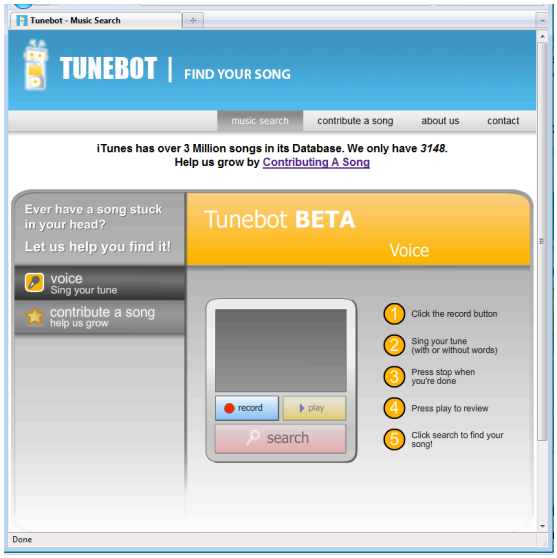


Figure 2. Screen shot of the Flash-based web interface for Tunebot.

The database for Tunebot uses searchable melodic keys derived from *a cappella* performances contributed by users as a result of playing Karaoke Callout, through use of the Tunebot search engine, and by logging in as a contributor and singing melodies to the system. Search keys are encoded as described in the section *Matching and Encoding*.

As of this writing, the Tunebot database contains roughly 11,000 examples for over 3,100 songs. Nearly 900 songs have 5 or more examples associated with them, and over 100 songs have 10 or more examples. The database is constantly growing as users contribute new songs and new examples for existing songs. To illustrate the rate of growth of the database, 5,053 examples representing 1,017 new songs were added to the database in the period from January 1, 2010 to April 15, 2010. At the current rate of growth, the size of the database should more than double by the end of this year compared to its size at the end of 2009.

System Overview

The Tunebot architecture is divided into three parts: (1) the client, (2) the server-side front-end, and (3) the

server-side back-end. These components are shown in Figure 3.

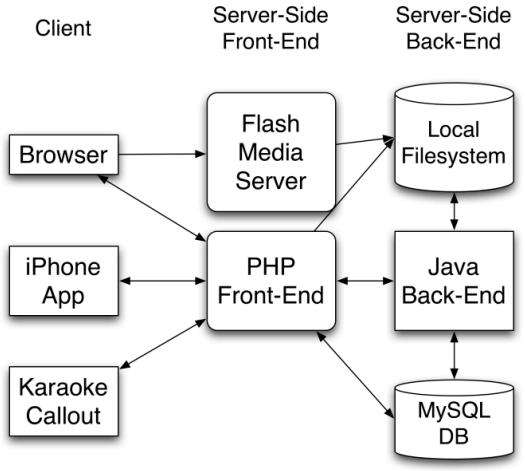


Figure 3. An overview of the Tunebot system.

The client-side is most typically a web browser. In this scenario, a Flash plug-in runs on the client side in the browser to record the audio of user queries and contributions and send it to the server.

The server-side front-end consists of two parts: (1) a set of PHP scripts served by an Apache Web Server, and (2) the Flash Media Server. The server-side front-end is responsible for presenting the user interface to search for and contribute songs, managing user information, and passing requests and audio files to the back-end. The iPhone client under development does not interact with the Flash Media Server, instead communicating with the server only through a PHP front end, as illustrated in Figure 3.

The server-side back-end is built around a Java servlet, running in Apache Tomcat. The back-end implements the matching algorithm and computes similarity rankings of submitted queries. Both the front and back ends interact directly with the SQL database on the server.

Encoding Melodies

Before a melodic comparison takes place, our transcriber estimates the fundamental frequency of the singing every 20 milliseconds. The note segmenter then divides this series of estimates into notes [18]. We encode all queries and all melodies in the database as sequences (strings) of note intervals. Each note interval is represented by a pair of values: the pitch interval (PI) between adjacent notes (measured in units of musical half-steps) and the log of the ratio between the length of a note and the length of the following note (LIR). Note lengths are defined to be inter-onset-intervals. We use note intervals encoded in this way because they are transposition invariant (melodies that differ only in key appear the same) and tempo invariant (melodies that differ only in tempo appear the same)g. We represent a melody $\boldsymbol{X}$ as a string of note intervals. The encoding of a sung example into note intervals is illustrated in Figure 4.

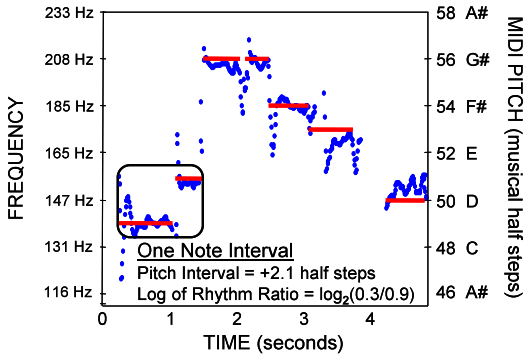


Figure 4. Pitch tracking and encoding of a sung example. Dots are pitch estimates. Horizontal lines are segmented notes. One note interval is shown in the rounded square.

Measuring Distance Between Melodies

Equation 1 defines a simple metric between note intervals $\boldsymbol{x}$ and $\boldsymbol{y}$, with pitch intervals $\boldsymbol{x_p}$ and $\boldsymbol{y_p}$ and LIRs $\boldsymbol{x_l}$ and $\boldsymbol{y_l}$.

$$d(x,y) = a|x_l - y_l| + b|x_p - y_p| \quad (1)$$

Here, $\boldsymbol{a}$ and $\boldsymbol{b}$ are non-negative weights chosen to optimize performance on a set of example queries for a given database of songs. Of course, when searching in a melodic database, one is not comparing individual note intervals, but full melodies. To compare melodic strings, we use edit distance [19].

The edit distance between two strings is the cost of the least expensive way of transforming one string into the other. Here, transformation cost (a.k.a. match cost) depends on the comparison function for the individual string elements described in Equation 1. We have a fixed insertion/deletion cost of one, effectively forcing the other parameters to be in these units.

This simple approach, when paired with a differential melodic encoding like our note-interval representation (this encoding is crucial to the use of such a simple note metric), has been shown to produce comparable search performance to more complex distance measures, without the need to optimize many parameters [4].

Each song in the database is represented by one or more sung melodies (search keys). A song's ranking in the search results is determined by the distance between the query and the nearest search key for that song.

Direct comparison of the query to every melody in the database becomes prohibitively slow as the size of the collection increases. If the comparison function for string elements is a metric (like Equation 1) then edit distance can also be made a metric [19]. Placing database melodies in a metric space allows efficient search

of a melodic database using vantage point trees [20, 21].

Vetting the Database

When a user queries for a particular song (e.g."Lola"), we consider a search successful if the correct target song is returned as one of the top answers. The closer the target gets to number one, the better the system performance. When a single search fails, it may be difficult to tell exactly why. The query may be poorly formed (singing "Hey Jude" when searching for "Lola"), the search method may be ineffective for a particular user (perhaps a user model needs optimization), or the individual search key may not correspond well with what a typical person would sing (storing only the verse when people sing only the chorus). Maintaining a database of past queries and their associated targets makes it possible to distinguish between cases and react appropriately.

Each row in Table 1 corresponds to a query made to a search engine. Here, "Query Audio" is the recorded singing, "Target Title" is the title of the correct target in the database and "Target Rank" is the rank of the correct target in the results returned by the system. In this example, every query by User 1 failed to place in the top ten. This is an indication that the search engine is not optimized properly for this user. Note also that every query for "Hey Jude" failed to place within the top fifty, regardless of user. This indicates a mismatch between the target and the kinds of query example users provide. This is in direct contrast to both "Que Sera Sera" and "Lola," each of which has one query whose correct target was ranked first.

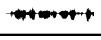
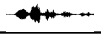
User	Query Audio	Target Title	Target Rank
1		Hey Jude	190
1		Que Sera Sera	39
1		Lola	21
2		Hey Jude	233
2		Lola	1
3		Hey Jude	59
3		Que Sera Sera	1

Table 1. Examples in a database

Our searchable database is composed of sung examples, keyed to correct song titles. This lets us automatically vet our search keys by using them as example queries. Those targets with below-average search results can then be tagged for search key updating. Such a database also allows for principled, automatic improvement of our similarity measures, as described in the section *System Optimization.*

System Optimization

Recall that searchable keys in the database are generated from past queries, sung contributions and examples of singing from Karaoke Callout. Each sung example is a potential new search key. The effectiveness of this new key can be measured by rerunning saved queries against this new key. This can be repeated using a key based on each query (or even on the union of all queries) and the best new key may then replace or augment the original search key for a particular song. This allows automatic, constant updating and improvement of the database without need for expert intervention.

A primary measure our system optimizes is mean reciprocal right rank (MRRR), shown in Equation 2. The right rank of a query is the rank of the correct song for the query. We refer to the correct song as the *target*. The mean right rank for a trial is the average right rank for all queries in the set.

$$MRRR = \frac{\sum_{n=1}^{numberOf\ Queries} \frac{1}{rightRank_n}}{numberOf\ Queries} \qquad (2)$$

We use MRRR because it gives more useful feedback than the simple mean right rank. Consider the following example. System A returns right ranks of 1, 1, 199, and 199 for four queries. System B returns 103, 102, 98, and 97. We prefer a system that ranks the correct target 1st half of the time to one that ranks it around 100th every time. Mean right rank returns a value of 100 for both systems. MRRR returns 0.5 for system A and 0.01 for system B.

When vetting search keys, one need only measure reciprocal right rank for each search key in the database. When this falls below a given value, it becomes a candidate for removal or replacement, as described above.

Similarly, we use MRRR as the measure of the effectiveness of a melodic similarity measure. We currently use the simple edit-distance melody metric described in a previous section because it allows the application of vantage-point trees to speed search. This metric, however, does have tunable parameters that let us weigh the relative importance of pitch and rhythm in melody matching. Our system allows re-tuning of the weight of such parameters after deployment, as the composition of the database and the user queries shift over time [18].

The relative importance of rhythm and pitch are characterized by the parameters *a* and *b*, respectively, in Equation 1. The *rhythm weight* is *a*, and *b* is referred to as the *pitch weight*. We cannot know *a priori* what values should be given to these parameters, so these values must be determined empirically. It also seems natural to wonder if different values would be appropriate for different individuals, depending on how accurate a given individual's singing is with regard to rhythm or pitch.

To explore these issues we first determined generally applicable values for these parameters by optimizing MRRR with respect to these parameters over a subset of the database. This process yielded an optimal value of 0.5 for rhythm weight and 0.005 for pitch weight. (These values only have meaning relative to the corresponding units used in Equation 1.) These values were set as the default parameter values of the system.

Next we collected a large number of labeled queries for a set of four heavy users of the system (more than 512 queries per user) and computed MRRR over a wide range of rhythm weight and pitch weight values. This served two purposes: to validate our choice of default parameter values, and to determine the importance of tuning these parameters per user. Note that this second set of queries, and the users who provided them, were not part of the initial optimization of the parameter values and so this constitutes a proper validation. Table 2 contains an illustrative excerpt of the analysis.

User	Best Rhythm Weight	Best Pitch Weight	Best Individual MRRR	% MRRR change from best global settings
1	0.400	0.00450	0.4760	+2.1%*
2	0.475	0.00450	0.4412	+0.9%*
3	0.525	0.00425	0.4065	+2.4%*
4	0.475	0.00500	0.3771	+2.4%*

Table 2. Optimal pitch and rhythm weights. Here, * means *not statistically significant*.

Each row of the table shows the result of optimizing MRRR with respect to rhythm weight and pitch weight for the given user. In each case the optimization was done over 512 labeled queries using a grid search with 17 points in each dimension and a granularity of 0.025 for rhythm weight and 0.00025 for pitch weight. This gave a total of 289 parameter value pairs tried for each user. The % change in MRRR is measured with respect to the MRRR achieved using the default rhythm and pitch weights learned from an earlier set of singers and examples.

The values shown for MRRR are based on an early 2010 snapshot of our constantly-growing database of real-world, user-contributed sung examples. For this experiment, all contributions from the singer for whom we optimize the values were removed prior to testing, as were all anonymous contributions to the database. This was done to ensure no contributions by the singer in question were used as searchable targets. Therefore the size of the test database depends on the number of contributions by the singer in question. The MRRR reported for User 1 was based on the largest resulting data set (5302 contributions representing 1919 unique songs). The data set for User 3 was the smallest (4556 contributions representing 1730 unique songs).

Several observations are possible from this table. The parameter values that result from optimizing per user are fairly close to the defaults learned from a large set of earlier singers. The optimal rhythm weight is within one grid point in three of four cases and the optimal pitch weight is within two grid points in three of four cases. More importantly, the improvement in MRRR from optimizing these parameters is quite small. In fact, it is less than 3% of the MRRR for each singer when using the default global parameter values learned from another set of singers. This difference is not statistically significant when taking into account the variance of MRRR on a random sample of 512 queries.

On the basis of this data and on similar analysis of other users, we are confident that our empirically determined global defaults for rhythm and pitch weights are valid and robust across a wide range of users, in the context of the current algorithm and the current composition of the database. Given the robustness of the default settings it appears that personalization in this parameter space is not necessary. However, our system contains several other parameter spaces and algorithmic choices where the importance of personalization has not yet been explored.

KARAOKE CALLOUT

In order to bootstrap the creation of a paired singing-example/target database and encourage user participation, we take a page from recent work in participatory and collaborative tagging. Particular inspirations include Open Mind [17] the ESP Game [16] and Karaoke Revolution (a popular video game released by Konami for the Sony PlayStation 2).

These examples have inspired us to cast system training in the form of a prototype interactive, client-server karaoke game: *Karaoke Callout.* This game closes the loop between system use and system improvement by providing correct song labels for sung examples, so that we can automatically vet and update a musical search engine.

An initial prototype of this game was originally developed for Symbian-OS phones [15]. Since creating the initial Symbian prototype, we have developed a new iPhone version of the game that is in beta testing with a small group of users. Those interested in becoming testers or in receiving notification of the final release of the game are encouraged to contact the authors of this paper.

The Karaoke Callout Game Interaction

The flow of Karaoke Callout proceeds as follows. Player 1 selects a song from our constantly growing database and sings it into the phone. While singing, the player is provided the lyrics to the song (Figure 5, step 1). If the player has the selected song in their iPod music library, then they have the option to sing along as the original recording plays.

Once the player is done singing, the audio is sent to the Tunebot music search engine, which rates the quality of the singing by measuring how closely it resembles the nearest melodic key for that song in the server database, sending a score back to the user (Figure 5, step 2)

Player 1 may then challenge another person to beat their score. If that person is a registered Karaoke Callout user, Player 1 needs to only provide the callout recipient's username, and they will be notified of the challenge via a push notification on their phone (Figure 5, step 3). If Player 1 wishes to invite a new person to play, they can select any email address (their phone contact list is provided as a convenience) and a mail will be sent to that person explaining how to install and play Karaoke Callout.

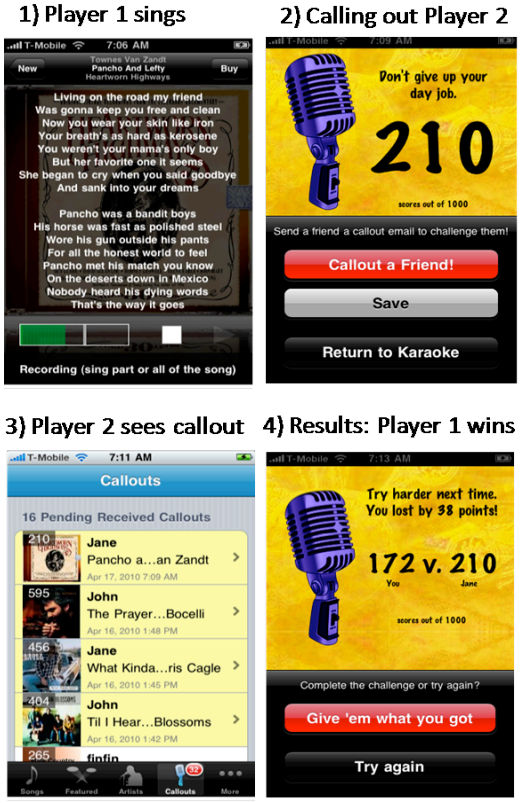


Figure 5. Screen shots of the iPhone interface for Karaoke Callout.

To accept the challenge, the callout recipient (Player 2) sings the song, attempting to better the performance of the challenger. The players are then notified of the results (Figure 5, step 4). This process may then be repeated, with either party selecting a new song with which to "call out" the other party. Over the course of an interaction, numerous examples of each party's singing are created and stored in our database.

Karaoke Callout System Architecture

The game server (see Figure 6) is divided into three main components. The first of these is the Karaoke Server (written in PHP), which handles communication with the clients, queries the Singing Scorer (our music search engine) and stores sung examples in the database. The final component is a SQL database of user accounts, audio queries, scores, and challenges. In addition to our server, the Apple Push Notification Service is also in the loop in order to communicate with the users when the game is not running. The Singing Scorer is modular and separate from the Karaoke Server, allowing each component to be updated independently. This is key for implementing automated system personalization and learning, as the Singing Scorer is the search engine that we wish to optimize (Tunebot).

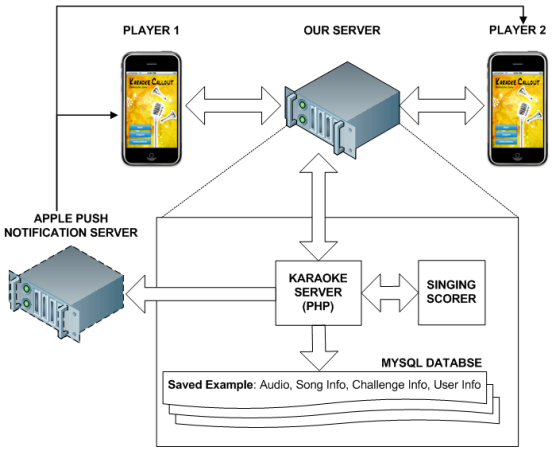


Figure 6. An overview of the KaraokeCallout system architecture.

USAGE STATISTICS

An ongoing goal of the Tunebot project has been to create a live, real-world system available to the general public, containing a growing set of songs that are of interest to a wide audience, and developed using data that represents the queries that real users generate. The usage statistics that follow were collected courtesy of Google Analytics.

In the period from January 15, 2010 to April 15, 2010, the Tunebot website had 15,421 unique visitors from 118 countries and territories. While more than three-quarters of these visits are from the United States and Canada, nearly 2,500 are from Europe and another 1,000 are from the rest of the world. The site receives between 100 and 200 hits on a typical day, most of which are new visitors. Figure 7 shows a breakdown of visitors by country of origin for the top ten countries.

Tunebot currently has more than 70 users who have chosen to register so that they may contribute songs to the system. It is clear the vast majority of users currently use the system anonymously to perform queries. We expect that broad dissemination of Karaoke Callout should increase the proportion of registered users.

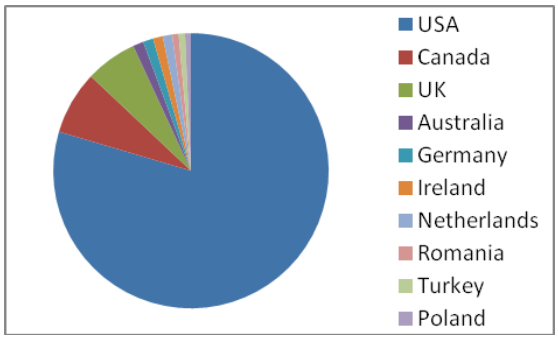


Figure 7. Proportion of visitors to Tunebot, by country of origin. Data collected over the period January 15 to April 15, 2010 (out of a total 15,421 unique visitors).

ANALYSIS AND CHALLENGES

Because we are developing a real-world system, some of our efforts have been directed at dealing with the practical issues that arise in implementing such a system, including robustness, scalability, efficiency, and system responsiveness.

The median length of a user query is around 18 seconds of audio, and our system currently takes about 5 seconds to return results from the time the query is received. For comparison, the longest query received to date is around 48 seconds long, and our system currently takes about 13 seconds to return a response to that query. The turnaround time is a function of several factors, including the size of the database and the length of the query, both in terms of the overall duration of the audio and the number of notes the user has sung. In the current implementation of the matching algorithm the running time is $O(kn)$, where k is the length of the query in notes and n is the number of keys in the database. While query lengths are not likely to change in the future, the size of the database is expected to grow dramatically over time. Algorithmic optimizations such as vantage point trees (discussed earlier) are one way to deal with the increasing query turnaround time. Another possibility, which we have implemented in our development environment but not yet in the production system, is to distribute the search algorithm across processors and compute matches in parallel. The potential for parallelization to speed up QBH is illustrated in [22].

Profiling analysis of our system has shown that a major portion of the query processing time is currently spent converting the raw audio of the query to the internal key representation, even though this phase of the algorithm does not dominate asymptotically. Future work includes exploring algorithmic and code-level optimizations to improve the running time of this portion of the algorithm.

A separate but related area of work has been to improve the scalability of our system in response to growing and fluctuating demand. This requires that the Tunebot service run on multiple machines concurrently, while maintaining a synchronized view of the database (so that, for example, a newly contributed song will be visible immediately to the user who contributed it). Cloud computing is an appealing solution to provide online services in a scalable and distributed fashion. We have developed a working prototype of Tunebot that is deployed as a virtual machine image on the Amazon Web Services cloud infrastructure.

MOVING FORWARD

We expect this work will lead to new insight into the mappings between human perception, human music production and machine-measurable features of music, as well as leading to new approaches to automatically tagging large databases of multimedia content, new approaches to individualized search engines for improved results and new approaches to speed multimedia search.

ACKNOWLEDGEMENTS

We would like to thank the National Science Foundation for funding to do this research. This work was supported by NSF Grant number IIS-0812314.

REFERENCES

[1] Haitsma, J. and T. Kalker. A Highly Robust Audio Fingerprinting System. in ISMIR 2002. 2002. Paris, France.

[2] Wang, A. An Industrial Strength Audio Search Algorithm. in 4th International Conference on Music Information Retrieval (ISMIR 2003). 2003. Baltimore, Maryland, USA.

[3] Typke, R., F. Wiering, and R.C. Veltkamp. A Survey of Music Information Retrieval Systems. in ISMIR 2005: 6th International Conference on Music Information Retrieval. 2005. London, England.

[4] Dannenberg, R., W. Birmingham, B. Pardo, N. Hu, C. Meek, and G. Tzanetakis, A Comparative Evaluation of Search Techniques for Query-by-Humming Using the MUSART Testbed. Journal of the American Society for Information Science and Technology, 2007: p. in press.

[5] Hewlett, W.B. and E. Selfridge-Field, eds. Melodic Similarity: Concepts, Procedures, and Applications. Computing in Musicology. Vol. 11. 1998, MIT Press: Cambridge, MA.

[6] Hu, N., R. Dannenberg, and A. Lewis. A Probabilistic Model of Melodic Similarity. in International Computer Music Conference (ICMC). 2002. Goteborg, Sweden: The International Computer Music Association.

[7] McNab, R.J., L.A. Smith, D. Bainbridge, and I.H. Witten, The New Zealand Digital Library MELody inDEX. D-Lib Magazine, 1997. May Issue.

[8] Uitdenbogerd, A. and J. Zobel. Melodic Matching Techniques for Large Music Databases. in Seventh ACM International Conference on Multimedia. 1999. Orlando, FL.

[9] Kornstadt, A., Themefinder: A Web-based Melodic Search Tool, in Melodic Similarity Concepts, Procedures, and Applications,, W. Hewlett and E. Selfridge-Field, Editors. 1998, MIT Press: Cambridge, MA.

[10] Gillet, O. and G. Richard, Drum Loops Retrieval from Spoken Queries. Journal of Intelligent Information Systems, 2005. 24(2-3): p. 159-177.

[11] Salamon, J. and M. Rohrmeier, A Quantitative Evaluation of a Two Stage Retrieval Approach for a Melodic Query by Example System, Proceedings of the 10th International Society of Music Information Retrieval Conference (ISMIR 2009), Kobe, Japan, 26-30 October 2009,

[12] Meek, C. and W. Birmingham, A Comprehensive Trainable Error model for sung music queries. Journal of Artificial Intelligence Research, 2004. 22: p. 57-91.

[13] Pauws, S. CubyHum: A Fully Operational Query by Humming System. in ISMIR 2002. 2002. Paris, France.

[14] Unal, E., S.S. Narayanan, H. Shih, E. Chew, and C.J. Kuo. Creating Data Resources for Designing User-centric Front-ends for Query by Humming Systems. in Multimedia Information Retrieval. 2003.

[15] Shamma, D. and B. Pardo. Karaoke Callout: using social and collaborative cell phone networking for new entertainment modalities and data collection, in Proceedings of ACM Multimedia Workshop on Audio and Music Computing for Multimedia (AMCMM 2006). 2006. Santa Barbara, CA, USA.

[16] von Ahn, L. and L. Dabbish. Labeling Images with a Computer Game. in CHI 2004. 2004. Vienna, Austria.

[17] Singh, P., The public acquisition of commonsense knowledge, in Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access. 2002, Palo Alto, CA.

[18] Little, D., Raffensperger, D, and B. Pardo, A Query by Humming System that Learns from Experience, Proceedings of the 8th International Conference on Music Information Retrieval, 2007 Vienna,Austria

[19] Wagner, R. and M. Fischer, The string-to-string correction problem. Journal of the ACM, 1974. 21(1): p. 168-173.

[20] Chavez, E., G. Navarro, and J.L. Marroquin, Searching in Metric Spaces. ACM Computing Surveys, 2001. 33(3): p. 273-321.

[21] Skalak, M., J. Han, B. Pardo, Speeding Melody Search with Vantage Point Trees, Proceedings of the International Society of Music Information Retrieval Conference(ISMIR 2008), Philadelphia, PA, USA, September 14-18, 2008.

[22] Ferraro, P., P. Hanna, L. Imbert, T. Izard, Accelerating Query-by-Humming on GPU, Proceedings of the International Society for Music Information Retrieval, 2009.

CONCURRENT CONSTRAINTS CONDITIONAL-BRANCHING TIMED INTERACTIVE SCORES

Mauricio Toro-Bermúdez **Myriam Desainte-Catherine**
Labri/ Université de Bordeaux 1
mtoro@labri.fr myriam@labri.fr

ABSTRACT

Multimedia scenarios have multimedia content and interactive events associated with computer programs. Interactive Scores (IS) is a formalism to represent such scenarios by temporal objects, temporal relations (TRs) and interactive events. IS describe TRs, but IS cannot represent TRs together with conditional branching. We propose a model for conditional branching timed IS in the Nondeterministic Timed Concurrent Constraint (ntcc) calculus. We ran a prototype of our model in Ntccrt (a real-time capable interpreter for ntcc) and the response time was acceptable for real-time interaction. An advantage of ntcc over Max/MSP or Petri Nets is that conditions and global constraints are represented declaratively.

1. INTRODUCTION

Interactive multimedia deals with the design of scenarios where multimedia content and interactive events can be associated with computer programs. Designers usually create multimedia for their scenarios, then they bind them to external interactive events or programs. *Max/MSP* and *Pure Data (Pd)* [1] are often used to program interactive scenarios. However, we claim for the need of a general model to (i) control synthesis based on human gestures and to (ii) declare relations among multimedia objects (e.g., partial-order relations for their execution).

Interactive Scores (IS) is a formalism for the design of scenarios represented by *temporal objects (TOs)*, *temporal relations (TRs)* and interactive events. Examples of TOs are videos and sounds. TOs can be triggered by interactive events (usually launched by the user) and several TOs can be active simultaneously. A TO can contain other TOs. The hierarchy allows us to control the start or end of a TO by controlling the start or end of its parent. Moreover, TRs provide a partial order for the execution of the TOs: TRs can be used to express precedence between objects.

IS have been subject of study since the beginning of the century [2], [3]. IS were originally developed for interactive music scores. Recently, the model was extended by Allombert, Desainte-Catherine, Larralde and Assayag in [4]. Hence IS can describe any kind of TOs, Allombert *et al.*'s model has inspired two applications: *iScore* [5] to compose and perform Electroacoustic music and *Virage* [6] to control live spectacles and interactive museums.

IS are successful to describe TRs, but IS have not been used to represent TRs together with *conditional branching*. Conditional branching is used in programming to describe control structures such as *if/else* and *switch/case*. It provides a mechanism to choose the state of a program depending on a condition and its current state.

Using conditional branching, a designer can create scenarios with loops and choices (as in programming). The user and the system can take decisions during performance with the degree of freedom described by the designer – while the system maintains the TRs of the scenario.

The designer can express under which conditions a loop ends; for instance, when the user changes the value of a certain variable, the loop stops; or the system non-deterministically chooses to stop.

Unfortunately, there is neither a theoretical model nor a special-purpose application to support conditional branching in interactive multimedia. In this work, we propose a model for conditional-branching timed IS in the *Nondeterministic Timed Concurrent Constraint* (`ntcc`) [7] calculus. In our model we combine TRs, conditional branching and discrete interactive events in a single model. We ran a prototype of the model over *Ntccrt* [8], a real-time capable interpreter for `ntcc`.

In a previous work [9], we showed how we can represent a multimedia installation with loops and choice [1], and the pure timed IS model [4] into our model.

1.1 Related work on interactive multimedia

A similar approach to ours was followed by Olarte and Rueda in [10]. They propose a model for IS in a calculus similar to `ntcc`; however, they only modeled TRs. They verified critical properties on the system. The key point of their model is that the user can change the hierarchical structure of the score during performance.

Another system dealing with a hierarchical structure is *Maquettes of OpenMusic* [11]. However, OpenMusic is a software for composition and not real-time interaction.

Another kind of systems capable of real-time interaction are *score following* systems (see [12]). Such systems track the performance of a real instrument and they may play multimedia associated to certain notes of the piece. However, to use these systems it is necessary to play a real

[1] `http://www.gmea.net/activite/creation/2007_2008/pPerez.htm`

instrument; whereas to use IS, the user only has to control some parameters of the piece, such as the start and end dates of the TOs.

A model for multimedia interaction that does not require a real instrument uses Hidden Markov Models to model probabilistic installations [13]. The system tracks human motion and it responds to human performance with chords and pitches depending on the knowledge of previous training. However, the system requires intensive training and it is not a tool for composition.

In the domain of composition of interactive music, there are applications such as *Ableton Live* [2] . Using *Live*, a composer can write loops and a musician can control different parameters of the piece during performance. Live is commonly used for Electronic and Electroacoustic music. Unfortunately, the means of interaction and the synchronization patterns provided by Live are limited.

1.1.1 Formalisms for Interactive Multimedia

To handle complex synchronization patterns and to predict the behavior of interactive scenarios, formalisms such as ntcc and *Hierarchical Time Stream Petri Networks (HTSPN)* [14] and have been used to model IS [15, 4].

In HTSPN we can express a variety of TRs, but it is not easy to represent global constraints (e.g., the number of TOs playing simultaneously). Instead, ntcc synchronizes processes through a common *constraint store*, thus global constraints are explicitly represented in such store. We chose ntcc because we can easily represent time, constraints, choice, and we can verify the model.

Another formalism for defining declaratively partial orders of musical processes and audio is Tempo [16]. However, Tempo does not allow us to express choice (when multiple conditions hold), simultaneity and weak time-outs (e.g., perform an action if the condition cannot be deduced). A key aspect is that there is a real-time capable interpreter and automatic verification for Tempo.

At present, there is not an automatic verifier for ntcc. In the declarative view, ntcc processes can be interpreted as *linear temporal logic* formulae. Ntcc includes an inference system in this logic to verify properties of ntcc models. This inference procedure was proved to be of exponential time complexity [17]. Nevertheless, we believe practical automatic verification could be envisioned for useful subsets of ntcc via model checking (see [18]).

Automated verification for IS will provide information about the correctness of the system to computer scientists. It will also provide important properties about the scenario to its designers and users. It will be possible to verify the absence of deadlocks, and also that certain TOs will be played during performance. This kind of properties cannot be verified in applications with no formal semantics.

1.2 Structure of the paper

The remainder of this paper is structured as follows. Section 2 explains ntcc and Ntccrt. Section 3 states our model for conditional-branching timed IS. Section 4 shows the ntcc definitions of our model. Section 5 explains our implementation using Pd and Ntccrt. Finally, section 6 gives some concluding remarks and future work.

[2] http://www.ableton.com/live/

2. THE NTCC PROCESS CALCULUS

A family of process calculi is *Concurrent Constraint Programming* (ccp) [19], where a system is modeled in terms of variables and constraints over some variables. The constraints are contained in a common *store*. There are also agents that reason about the system variables, based on partial information (by the means of constraints).

Formally, ccp is based upon the idea of a *constraint system (CS)*. A constraint system includes a set of (basic) constraints and a relation (i.e., entailment relation $\models$) to deduce a constraint with the information supplied by other constraints.

A ccp system usually includes several CSs for different variable types. There are CSs for variable types such as sets, trees, graphs and natural numbers. A CS providing arithmetic relations over natural numbers is known as *Finite Domain (FD)*. As an example, using a FD CS, we can deduce $pitch \neq 60$ from the constraints $pitch > 40$ and $pitch < 59$.

Although we can choose an appropriate CS to model any problem, in ccp it is not possible to delete nor change information accumulated in the store. For that reason it is difficult to perceive a notion of discrete time, useful to model reactive systems communicating with an external environment (e.g., users, lights, sensors and speakers).

Ntcc introduces to ccp the notion of discrete time as a sequence of *time units*. Each time unit starts with a store (possibly empty) supplied by the environment, and ntcc executes all the processes scheduled for that time unit. In contrast to ccp, in ntcc we can model variables changing values over time. A variable x can take different values at each time unit. To model that in ccp, we have to create a new variable x_i each time we change the value of x.

2.1 Ntcc in multimedia interaction

In this section we give some examples on how the computational agents of ntcc can be used with a FD CS. A summary of the agents semantics can be found in Table 1.

Agent	Meaning
tell (c)	Adds c to the current store
when (c) **do** A	If c holds now run A
local (x) **in** P	Runs P with local variable x
$A \parallel B$	Parallel composition
next A	Runs A at the next time-unit
unless (c) **next** A	Unless c holds, next run A
$\sum_{i \in I}$ **when** (c_i) **do** P_i	Chooses P_i s.t. (c_i) holds
$*P$	Delays P indefinitely
$!P$	Executes P each time-unit

Table 1. Semantics of ntcc agents.

- Using *tell* it is possible to add constraints to the store such as **tell**$(60 < pitch_2 < 100)$, which means that $pitch_2$ is an integer between 60 and 100.

- *When* can be used to describe how the system reacts to different events; for instance, **when** $pitch_1 = C4 \wedge pitch_2 = E4 \wedge pitch_3 = G4$ **do tell**$(CMayor = true)$ adds the constraint $CMayor = true$ to the current store as soon as the pitch sequence C, E, G has been played.

- *Parallel composition* ($\|$) makes it possible to represent concurrent processes; for instance, **tell** $(pitch_1 = 52) \parallel$ **when** $48 < pitch_1 < 59$ **do tell** $(Instrument = 1)$ tells the store that $pitch_1$ is 52 and concurrently assigns the *instrument* to one, since $pitch_1$ is in the desired interval (see fig. 1).

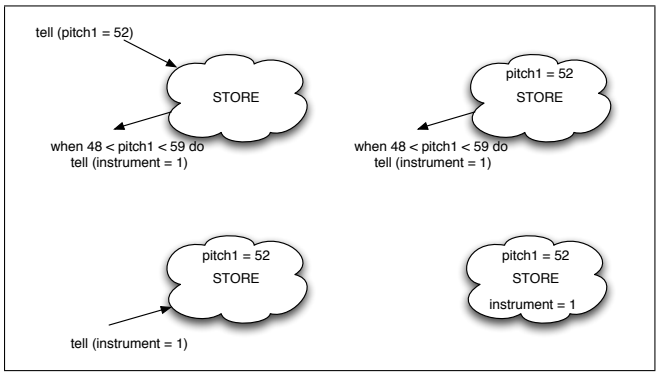


Figure 1. An example of the `ntcc` agents.

- *Next* is useful when we want to model variables changing over time; for instance, **when** $(pitch_1 = 60)$ **do next tell** $(pitch_1 <> 60)$ means that if $pitch_1$ is equal to 60 in the current time unit, it will be different from 60 in the next time unit.

- *Unless* is useful to model systems reacting when a condition is not satisfied or when the condition cannot be deduced from the store; for instance, **unless** $(pitch_1 = 60)$ **next tell** $(lastPitch <> 60)$ reacts when $pitch_1 = 60$ is false or when $pitch_1 = 60$ cannot be deduced from the store (e.g., $pitch_1$ was not played in the current time unit).

- *Star* (*) can be used to delay the end of a process indefinitely, but not forever; for instance, $*$**tell** $(End = true)$. Note that to model Interactive Scores we do not use the *star* agent.

- *Bang* (**!**) executes a certain process every time unit after its execution; for instance, !**tell** $(C_4 = 60)$.

- *Sum* ($\sum$) is used to model non-deterministic choices; for instance, $\sum_{i \in \{48,52,55\}}$ **when** $i \in PlayedPitches$ **do tell** $(pitch = i)$ chooses a note among those played previously that belongs to the C major chord.

In `ntcc`, recursion can be defined (see [17]) with the form $q(x) =^{def} P_q$, where q is the process name and P_q is restricted to call q at most once and such call must be within the scope of a *next*. The reason of using *next* is that `ntcc` does not allow recursion within a time unit.

The reader should not confuse a simple definition with a recursive definition; for instance, $Before_{i,j} =^{def}$ **tell**$(i \in Predecessor_j)$ is a simple definition where the values of i and j are replaced statically, like a macro in a programming language. Instead, a recursive definition such as $Clock(v) =^{def}$ **tell**$(clock = v) \| $**next** $Clock(v+1)$ is like a function in a programming language.

2.2 Ntccrt: A real-time capable interpreter for ntcc

In the current version of Ntccrt, we can write a `ntcc` model on either Lisp, Openmusic or C++. For a complete implementation of Interactive Scores, it will be necessary to produce automatically the corresponding `ntcc` model based on a graphical interface similar to Virage.

To execute a `ntcc` model it is not necessary to develop an interface because Ntccrt programs can be compiled into stand-alone programs or as external objects (i.e., a binary plugins) for Pd or Max (see fig. 2).

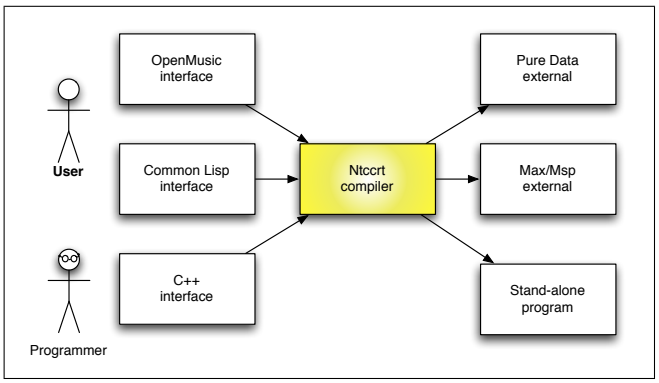


Figure 2. Interfaces of Ntccrt.

We can use the message passing API provided by Pd and Max to communicate any object with the Ntccrt external. We can also control all the available objects for audio and video processing defined in those languages using Ntccrt. To synchronize those objects, Ntccrt provides an important part of Gecode's constraints [20].

Ntccrt uses Gecode as its constraint solving library. Gecode was carefully designed to support efficiently the Finite Domain (FD) constraint system. Ntccrt relies on propagation of FD constraints.

3. CONDITIONAL BRANCHING TIMED IS

Points and intervals build up Interactive Scores (IS), thus a *score*[3] (i.e., the specification of a scenario) is defined by a tuple $s = \langle P, I \rangle$, where P is a set of points and I is a set of intervals. A temporal object is just a type of interval.

3.1 Points

Intuitively, a point p is a *predecessor* of q if there is a relation p *before* q. Analogically, a point p is a *successor* of r if there is a relation r *before* p.

A *Point* is defined by $p = \langle b_p, b_s \rangle$, where b_p and b_s represent the behavior of the point. Behavior b_p defines

[3] We still use the term *score* for historical reasons.

whether the point waits until all its predecessors transfer the control to it –*Wait for All (WA)*– or it only waits for the first of them –*Wait for the First (WF)*–. Behavior b_s defines whether the point transfers the control to all its successors which conditions hold –*No CHoice (NCH)*– or it chooses one of them –*CHoice (CH)*–.

Note that we do not include the set of dates of the point in previous definition. Beurivé *et al.* argued in [2] that the edition of a hierarchical representation of music using a relative time model requires less variable updates than using an absolute time model. We argue that it is also true during the performance of Interactive Scores. Moreover, in our model it is not easy to know the set of all possible dates *a priori* because they depend on the choices that the user makes during performance.

3.2 Intervals: TCRs and TOs

An interval p *before* q intuitively means that the system waits a certain time to transfer the control from p to q if the condition in the interval holds. In addition, it executes a process throughout its duration. An interval also has a *nominal duration* that may change during the performance. The nominal duration is computed during the edition of the scenario using constraint programming (see [15]. Formally, an interval is a tuple composed by

- a start point (p_1)
- an end point (p_2)
- a condition (c)
- a duration (d)
- an interpretation for the condition (b)
- a local constraint (l)
- a process $(proc)$
- parameters for the process $(param)$
- children (N)
- local variables $(vars)$

It is not practical to include all those elements explicitly; thus, we have identified two types of intervals. *timed conditional relations (TCRs)* have a condition c and an interpretation b, but they do not have children, their local constraint is `true`, and their process is $silence$[4]. *Temporal objects (TOs)* may have children, local variables and a local constraint, but their condition is `true`, and their interpretation is $when$ (i.e., when the condition is true, it transfers the control from p_1 to p_2).

To have a coherent score, we must define a TCR between the start point of each father and the start point of at least one of its children. However, it is not required to connect a child to the end point of its father. Furthermore, in our model we may define multiple TCRs and TOs between two points. This does not introduce an incoherence in the model because the behavior of those intervals (as any interval) depends on the behavior of the points and the parameters of the interval.

3.2.1 Timed Conditional Relations (TCRs)

A *timed conditional relation (TCR)* is defined by $r = \langle p_1, p_2, c, d, b\rangle$, where p_1 and p_2 are the points involved in the relation. The condition c determines whether the control *jumps* from p_1 to p_2 (i.e., the control is transferred from p_1 to p_2). The interpretation of c is b. There are two possible values for b: (i) *when* means that if c holds, the control jumps to p_2; and (ii) *unless* means that if c does not hold or its value cannot be deduced from the environment (e.g., $c = a > 0$ and $-\infty < a < \infty$), the control jumps to p_2.

A duration is *flexible* if it can take any value, *rigid* if it takes values between two fixed integers and *semi-rigid* if it takes values greater than a fixed integer. In our model, we always respect flexible durations. Our model is based upon transferring the control from one point to another. For that reason, it is not always possible to respect rigid and semirigid durations; for instance, when a point waits for an event or when it is followed by a choice.

3.2.2 Temporal objects (TOs)

A *temporal object (TO)* is defined by $t = \langle p_s, p_e, l, d, proc, param, N, vars\rangle$ where p_s is a point that starts a new instance of t and p_e ends such instance. A constraint l is attached to t, it contains local information for t and its children. The duration is d. A process which executes throughout the duration of t is $proc$. The list of parameters for the process is $param$. The set of TOs embedded in t is N, which are called children of t. Finally, $vars$ represents the local variables defined for the TO that can be used by t's children, process and local constraint.

3.3 Example: A loop controlled by a condition

The following example (see fig. 3) describes a score with a loop. During the execution, the system plays a silence of one second. After the silence, it plays the sound B during three seconds and simultaneously it turns on the lights D for one second. After the sound B, it plays a silence of one second, then it plays video C. If the variable $finish$ becomes true, it ends the scenario after playing the video C; otherwise, it jumps back to the beginning of the first silence after playing the video C.

To define the score of this scenario, we define a local boolean variable $finish$ in A, and we use it as the condition for some TCRs. Note that the silence between D and C lasts one second in the score, but during execution it is longer because of the behavior of the points.

The points have the following behavior. The end point of C (e_c) is enabled for choice, and the other points transfer the control to all their successors. The start point of C (s_c) waits for all its predecessors to transfer the control to it, and all the other points wait for the first predecessor that transfers the control to them.

Formally, the points are defined

$$s_a = e_a = s_b = e_b = s_d = e_d = \langle\{WF, NCH\}\rangle$$
$$s_c = \langle\{WA, NCH\}\rangle$$
$$e_c = \langle\{WF, CH\}\rangle$$
$$P = \{s_a, e_a, s_b, e_b, s_c, e_c, s_d, e_d\}$$

As an example, e_c Waits for the first predecessor (WF) and makes a choice (CH).

[4] $silence$ is a process that does nothing.

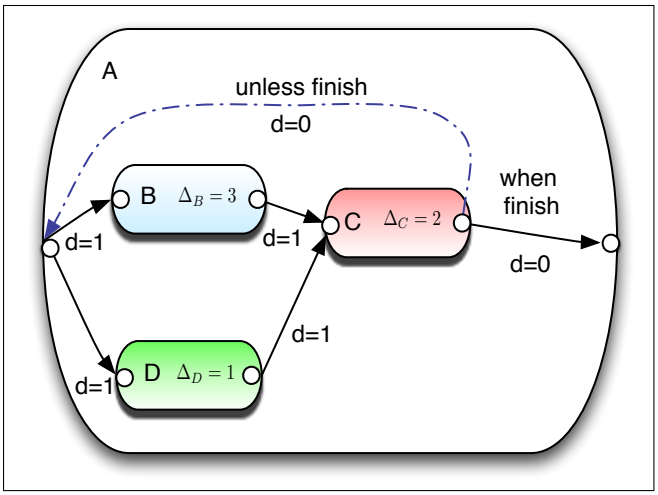


Figure 3. A score with a user-controlled loop.

The TOs are defined by

$A = \langle s_a, e_a, d \in [0, \infty), d, sil., \emptyset, \{B, C, D\}, \{finish\}\rangle$
$B = \langle s_b, e_b, \texttt{true}, 3, playSoundB, \emptyset, \emptyset, \emptyset\rangle$
$C = \langle s_c, e_c, \texttt{true}, 2, PlayVideoC, \emptyset, \emptyset, \emptyset\rangle$
$D = \langle s_d, e_d, \texttt{true}, 1, TurnOnLightsD, \emptyset, \emptyset, \emptyset\rangle$
$T = \{A, B, C, D\}$

As an example, A is composed by points s_a and e_a, it has a flexible duration, its process is silence, its children are B, C and D and its local variable is $finish$.

In what follows we present the TCRs

$TCR =$
$\{\langle s_a, s_b, \texttt{true}, 1, when\rangle, \langle s_a, s_d, \texttt{true}, 1, when\rangle,$
$\langle e_b, s_c, \texttt{true}, 1, when\rangle, \langle e_d, s_c, \texttt{true}, 1, when\rangle,$
$\langle e_c, s_a, \neg finish, 0, when\rangle, \langle e_c, e_a, finish, 0, when\rangle\}$

As an example, the first one is a TCR between points s_a and s_b, its condition is `true`, its interpretation is $when$ and its duration is one.

Finally, I is the set of intervals composed by the TOs and the TCRs and S is the score.

$I = T \bigcup TCR \qquad S = \{P, I\}$

3.4 Limitations: Rigid durations and choice

In some cases (e.g., fig. 3), we can respect rigid durations of TOs during performance. Unfortunately, there is not a generic way to compute the value of a rigid duration in a score with conditional branching. The problem is that choices do not allow us to predict the duration of a TO's successor; therefore, it is not possible to determinate *a priori* the duration of all the TOs.

Figure 4 shows a scenario where we cannot respect rigid durations. T_2, T_4 and T_5 have fixed durations, but T_1 can take different values between Δ_{min} and Δ_{max}. Since there is no way to predict whether T_2 or T_5 will be chosen after the execution of T_1, we cannot compute a coherent duration for T_1 before the choice.

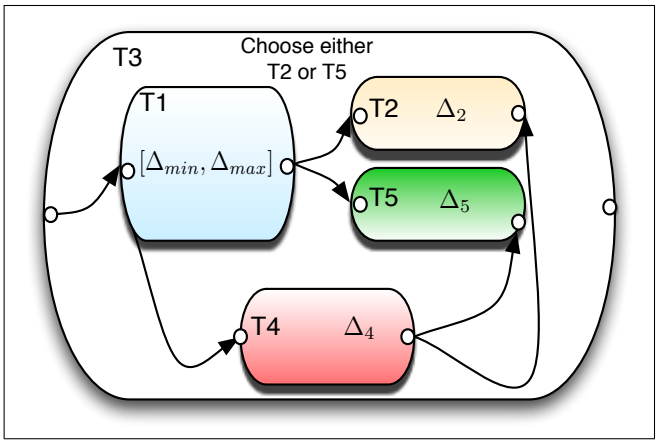


Figure 4. Limitation of rigid durations.

4. OUR NTCC MODEL OF IS

In this section we define our `ntcc` model. We define processes for some combinations of the behaviors of a point. The definition of an interval can be used for both timed conditional relations and temporal objects. To represent intervals we create a graph with the predecessors and successors of each point using the variables $Predec$ and $Succ$. For simplicity, we do not include hierarchy, we only model the interpretation $when$, we can only declare a single interval between two points, and we can only execute a single instance of an interval at the same time.

4.1 Points: Three combinations of behaviors

We only include three type of points: points that choose among their successors ($ChoicePoint$), points that transfer the control to all their successors ($JumpToAllPoint$), and points that wait for all their predecessors to transfer the control to them ($WaitForAllPoint$). The first two types of points wait for the first predecessor that transfers the control to them to be active.

Points are modeled using Finite Domain constraints; for instance, to know if at least one point has transferred the control to the point i, we ask to the store if the *boolean or* ($\bigvee_{j \in P}$) constraint applied to the relation $Arrived(i, j)$ can be deduced from the store (where P is the set of identifiers for each point).

When all the expected predecessors transfer the control to the point i, we say that the point is active (i.e., $ActivePoints_i$ holds). Analogaly, when a point i transfers the control to a point j, we add the constraint $ControlTranferred(j, i)$.

In order to represent the choice between points a and b, we use the variable $finish$ in the Σ process. Note that **when** c_1 **do** P_1 + **when** c_2 **do** P_2 is equivalent to $\sum_{i \in \{1,2\}}$ **when** c_i **do** P_i, and **whenever** c **do** P is equivalent to **!when** c **do** P.

$ChoicePoint_{i,a,b} \stackrel{def}{=}$
whenever $\bigvee_{j \in P} Arrived(i, j)$ **do** (**tell** $(ActivePoints_i)$
$\parallel$ **when** $finish$ **do tell** $(ControlTransferred(a, i))$

+**when** $\neg finish$ **do tell** $(ControlTransferred(b,i)))$

The following definition uses the agent $\prod$ to transfer the control to all the successors of the point i. The agent $\prod$ represents the parallel composition in a compact way.

$$ToAll_i \stackrel{def}{=} \textbf{tell}\ (ActivePoints_i) \ \|\textstyle\prod_{j\in P}\ \textbf{when}\ Succs(i,j)\ \textbf{do}\ \textbf{tell}\ (ControlTransferred(j,i)))$$

Using the definition $ToAll_i$, we define the two points that transfer the control to all its successors.

$$JumpToAllPoint_i \stackrel{def}{=} \textbf{whenever}\ \textstyle\bigvee_{j\in P} Arrived(i,j)\ \textbf{do}\ ToAll_i$$

To wait for all the predecessors, we ask the store if the constraint $Arrived = Predec$ holds.

$$WaitForAllPoint_i \stackrel{def}{=} \textbf{whenever}\ \forall j, Arrived(i,j) = Predec(i,j)\ \textbf{do}\ ToAll_i$$

4.2 Intervals: TCRs and TOs

Intervals are modeled by two recursive definitions. These definitions model both TOs and TCRs because intervals only change the value of an $ActivePoints$ variable, thus they only control the start and end of their processes.

Process I waits until at least one point transfers the control to its start point i, and at least one point has been chosen by another point to transfer the control to its destination j. When such conditions hold, it waits until the duration of the interval is over [5], then it transfers the control from point i to j. It also adds a constraint on the corresponding set of predecessors and successors.

$$I_{i,j,d} \stackrel{def}{=} !(\textbf{tell}\ (Predec(j,i)) \parallel \textbf{tell}\ (Succ(i,j))) \ \|\textbf{whenever}\ \textstyle\bigvee_{k\in P} ControlTransferred(j,k) \wedge \bigvee_{k\in P} Arrived(i,k)\ \textbf{do}(\ \textbf{next}^d(\textbf{tell}(Arrived(j,i)) \|PredecessorsWait(i,j)))$$

$PredecessorsWait$ adds the constraint $Arrived(j,i)$ until the time unit after the point j becomes active. This definition maintains the coherence of $WaitForAll$ points.

$$PredecessorsWait_{i,j} \stackrel{def}{=} \textbf{unless}\ ActivePoints_j\ \textbf{next}\ (PredecessorsWait_{i,j} \| \textbf{tell}\ (Arrived(j,i)))$$

4.3 The example 3.3 on ntcc

The example presented on figure 3 can be easily modeled in `ntcc`. $User$ is a process representing a user that tells to the store that $finish$ is not true during the first n time units, then it tells that $finish$ is true. Note that an advantage of `ntcc` is that the constraint $i \geq n$ can be easily replaced by more complex ones; for instance, it can be replaced by $i \geq n \wedge c$. Constraint c can be, for instance, "there are only three active points at this moment in the score" (i.e., $|\{x \in ActivePoints \mid x = 1\}| = 3$).

$$User_n(i) \stackrel{def}{=} \textbf{when}\ i \geq n\ \textbf{do tell}\ (finish) \ \|\textbf{unless}\ i \geq n\ \textbf{next tell}\ (\neg finish) \ \|\textbf{next}\ User_n(i+1)$$

$$TCRs \stackrel{def}{=} I_{s_d,e_d,1} \ \|I_{s_a,s_b,1}\|I_{e_d,s_c,1}\|I_{s_b,e_b,3}\|I_{e_b,s_c,1}\|I_{s_c,e_c,2}\|I_{e_c,s_a,0} \ \|I_{e_c,e_a,0}\|I_{null,s_a,0}\|I_{s_a,s_d,1}\|\ \textbf{tell}\ (Arrived(s_a,start))$$

$$Points \stackrel{def}{=} ChoicePoint_{e_c,e_a,s_a}\|WaitForAllPoint_{s_c} \ \|\textstyle\prod_{i\in\{s_a,e_a,s_b,e_b,s_d,e_d\}} JumpToAllPoint_i$$

$$System_n \stackrel{def}{=} User_n(0)\|TCRs\|Points$$

5. IMPLEMENTATION IN NTCCRT AND PD

We implemented the previous example in Ntccrt and Pure Data (Pd) (fig. 5). We replaced the $User$ process with a user input for the variable $finish$. We generated a Ntccrt external (i.e., a binary plugin) for Pd with our `ntcc` model.

The external has two inputs: one for the clock ticks and one for the value of $finish$. The input for the clock ticks can be connected to a *metronome* object to have a fixed duration for every time unit during the performance. The reader can find a discussion of executing time units with fixed durations in [8].

The Ntccrt external outputs a boolean value for each point, indicating whether it is active or not. Using such values, we can control the start and end of $SoundB$, $VideoC$ and $lightsD$, which are processes defined in Pd.

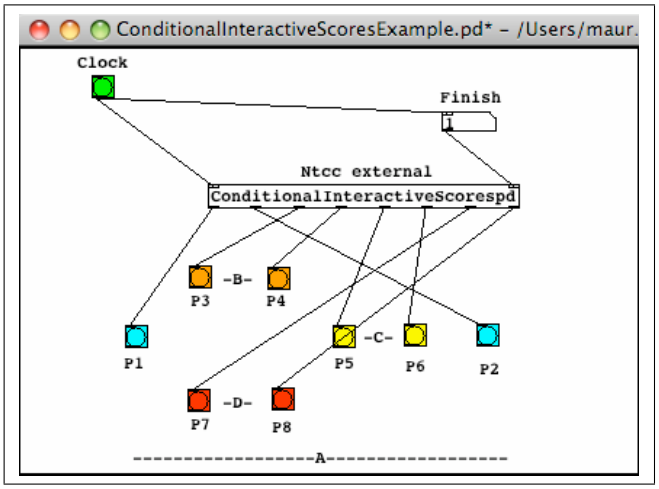


Figure 5. Executing Example 3.3 in Pd.

5.1 Results: Performance and usability of Ntccrt

We built automatically Interactive Scores (IS) with a number of points [6] and relations in the order of 2^n, with n from two to ten (see fig. 6). We ran each score 100 times as a stand-alone program. The duration of a time unit is determined by the time taken by Ntccrt to calculate the output, not by an external clock. The tests were performed on an

[5] *next*^{d} is a process *next* nested d times ($next(next(next...)$).

[6] The exact number of points is $3.2^n - 2$.

iMac 2.6 GHz with 2 GB of RAM under Mac OS 10.5.7. It was compiled with GCC 4.2 and liked to Gecode 3.2.2.

The authors of the Continuator [21] argue that a multimedia interaction system with a response time less than 30 ms is able to interact in real-time with even a very fast guitar jazz player. Therefore, our results (fig. 7) are acceptable for real-time interaction with a guitarist for up to 1000 points (around 500 TOs). We conjecture that a response time of 20 ms is appropriate to interact with a very fast percussionist. In that case, we can have up to 400 TOs.

5.1.1 Usability of Ntccrt

We found out intuitive to write `ntcc` models in Ntccrt, to someone familiar with `ntcc`, because it provides a Lisp interface with a syntax similar to `ntcc`; for instance, $PredecessorWait$ is written as

```
(defproc PredecessorsWait (i j)
 (unlessp (v=? (ActivePoint i) j)
  (||(call PredecessorsWait i j)
    (tell= (ArrivedPoint j i) 1))))
```

It is slightly harder to write the same definition in C++

```
class predecessorsWait:public proc{
public:
AskBody* predecessorsWait::operator()(
Space* h, vector<int> intparameters,
vector<variable *> variableparameters)
{return unless(eq(ActivePoint[i][j]),
parallel(call(PredecessorWait,i,j),
tellEqual(ArrivedPoint[i][j],1)));}};
```

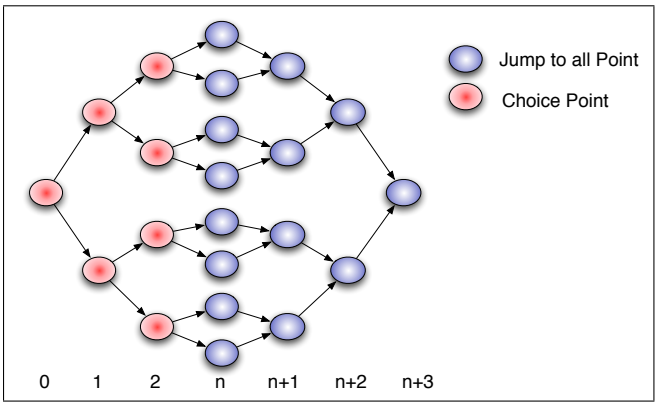


Figure 6. A scalable-size score with $3.2^n - 2$ points.

6. CONCLUDING REMARKS

We developed a model for multimedia interaction with conditional-branching and temporal relations based on points and intervals. We implemented it using Ntccrt and Pure Data (Pd). We conclude from performance results that our prototype is compatible with real-time interaction for a reasonable amount of points and relations. An existing implementation of Interactive Scores model is also capable of real-time and it can easy respect rigid durations, but such model does not support loops nor choice.

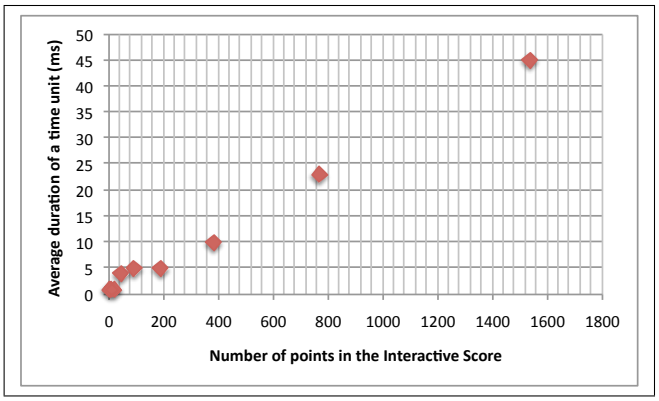


Figure 7. Performance of the simulation fo the score in Fig. 6.

For simplicity, in our prototype we do not include hierarchy, we only model the interpretation $when$, we can only declare a single interval between two points, we can preserve rigid durations only in a few cases , and we can only execute a single instance of an interval at the same time.

An advantage of `ntcc` with respect to previous models of Interactive Scores, Pd, Max and Petri Nets is representing declarative conditions by the means of constraints. Complex conditions, in particular those with an unknown number of parameters, are difficult to model in Max or Pd. To model generic conditions in Max or Pd, we would have to define each condition either in a new patch or in a predefined library. In Petri nets, we would have to define a net for each condition.

6.1 Future work

Ntccrt is not yet an interface for composers and designers of multimedia scenarios. For them is much more intuitive an interface such as Virage [6]. A graphical interface for our model should provide the means to specify the score as done in Example 3.3

Once we have the graphical interface, we plan to model audio processes in `ntcc` and replace them in the implementation by *Faust* programs [22] which also have formal semantics. Using Faust, we can gain efficiency and preserve the formal properties of our model (see [23] for a description of this idea).

7. ACKNOWLEDGEMENTS

We want to acknowledge Joseph Larralde and Camilo Rueda for their valuable comments on this model, and Andrés Porras for his editing contributions.

8. REFERENCES

[1] M. Puckette, T. Apel, and D. Zicarelli, "Real-time audio analysis tools for Pd and MSP," in *Proc. of ICMC '98*, 1998.

[2] A. Beurivé and M. Desainte-Catherine, "Representing musical hierarchies with constraints," in *7th International Conference on Principles and Practice of Con-*

straint Programming, Musical Constraints Workshop, Paphos, 2001.

[3] M. Desainte-Catherine and N. Brousse, "Towards a specification of musical interactive pieces," in *Proc. of of CIM XIX, Firenze, Italy.*, 2003.

[4] A. Allombert, M. Desainte-Catherine, J. Larralde, and G. Assayag, "A system of interactive scores based on qualitative and quantitative temporal constraints," in *Proc. of Artech 2008*, 2008.

[5] A. Allombert, G. Assayag, and M. Desainte-Catherine, "Iscore: a system for writing interaction," in *Proc. of DIMEA '08*, (New York, NY, USA), pp. 360–367, ACM, 2008.

[6] A. Allombert, P. Baltazar, R. Marczak, M. Desainte-Catherine, and L. Garnier, "Designing an interactive intermedia sequencer from users requirements and theoretical background," in *Proc. of ICMC 2010*, 2010.

[7] M. Nielsen, C. Palamidessi, and F. Valencia, "Temporal concurrent constraint programming: Denotation, logic and applications," *Nordic Journal of Comp.*, vol. 1, 2002.

[8] M. Toro-B., C. Agón, G. Assayag, and C. Rueda, "Ntcrt: A concurrent constraint framework for real-time interaction," in *Proc. of ICMC '09*, 2009.

[9] M. Toro-B., M. Desainte-Catherine, and P. Baltazar, "A model for interactive scores with temporal constraints and conditional branching," in *Proc. of Journées d'informatique musical (JIM) '10*, May 2010.

[10] C. Olarte and C. Rueda, "A Declarative Language for Dynamic Multimedia Interaction Systems," *Mathematics and Computation in Music*, vol. 38, 07 2009.

[11] J. Bresson, C. Agón, and G. Assayag, "Openmusic 5: A cross-platform release of the computer-assisted composition environment," in *10th Brazilian Symposium on Computer Music*, 2005.

[12] A. Cont, "Antescofo: Anticipatory synchronization and control of interactive parameters in computer music," in *Proc. of ICMC '08*, 2008.

[13] C. G. Baltera, S. B. Smith, and J. A. Flanklin, "Probabilistic interactive installations," in *Proc. of FLAIRS Conference '07*, pp. 553–558, 2007.

[14] P. Sénac, P. d. Saqui-Sannes, and R. Willrich, "Hierarchical time stream petri net: A model for hypermedia systems," in *Proc. of the 16th International Conference on Application and Theory of Petri Nets*, (London, UK), pp. 451–470, Springer-Verlag, 1995.

[15] A. Allombert, G. Assayag, M. Desainte-Catherine, and C. Rueda, "Concurrent constraint models for interactive scores," in *Proc. of SMC '06*, May 2006.

[16] R. Ramirez, "A logic-based language for modeling and verifying musical processes," in *Proc. of ICMC '06*, 2006.

[17] F. D. Valencia, *Temporal Concurrent Constraint Programming*. PhD thesis, University of Aarhus, 2002.

[18] M. Falaschi and A. Villanueva, "Automatic verification of timed concurrent constraint programs," *Theory Pract. Log. Program*, vol. 6, no. 4, pp. 265–300, 2006.

[19] V. A. Saraswat, *Concurrent Constraint Programming*. MIT Press, 1992.

[20] G. Tack, *Constraint Propagation - Models, Techniques, Implementation*. PhD thesis, Saarland University, Germany, 2009.

[21] F. Pachet, "Playing with virtual musicians: the continuator in practice," *IEEE Multimedia*, vol. 9, pp. 77–82, 2002.

[22] Y. Orlarey, D. Fober, and S. Letz, "Syntactical and semantical aspects of faust," *Soft Comput.*, vol. 8, no. 9, pp. 623–632, 2004.

[23] M. Toro-B., "Structured musical interactive scores (short)," in *Proc. of the International Computer Logic Programming (ICLP '10) (To appear)*, 2010.

D-JOGGER: SYNCING MUSIC WITH WALKING

Bart Moens
IPEM, Dept. of Musicology
Ghent University, Belgium
bmmoens.moens@ugent.be

Leon van Noorden
IPEM, Dept. of Musicology
Ghent University, Belgium
leonvannoorden@mac.com

Marc Leman
IPEM, Dept. of Musicology
Ghent University, Belgium
marc.leman@ugent.be

ABSTRACT

We present D-Jogger, a music interface that makes use of body movement to dynamically select music and adapt its tempo to the user's pace. D-Jogger consists of several independent modules, such as a step detection algorithm and tempo-aware playlists, to achieve this goal. The research done with D-Jogger has focused on entrainment: the synchronization of two rhythmical processes, in this case music and walking. We present several ways of visualizing entrainment data, including synchronization plots and phase histograms.

A pilot experiment was performed using D-Jogger with 33 participants. Preliminary data suggest that, when the music's tempo and the user's pace are close enough to each other, most users synchronize their walking to the music - taking a step with each beat. A user survey indicated that participants experience this effect as stimulating and motivating.

Several other application domains for D-Jogger are possible: personal training devices for joggers, rehabilitation therapy for Parkinson patients or simply as a nice-to-have application for your mobile phone.

1. INTRODUCTION AND RELATED WORK

Entrainment is broadly defined as a phenomenon in which two or more independent rhythmic processes synchronize with each other [2]. In a more constrained sense, it is the synchronization of a system with a variable frequency to an external frequency. We see this process every day, for example when people dance to music, they perform rhythmical movements in sync with the beat – on the perceived pulse of the music.

Another form of entrainment can happen when people are walking to music. Previous research has shown that people can synchronize their walking movements with music over a broad range of tempi and that this synchronization is most optimal around 120 beats per minute (BPM) [13]. Moreover, in [11] it is speculated that this frequency of 2 Hz represents some form of central resonance of human movement.

This paper describes the framework of D-Jogger. The framework is used to create applications that make use of the body movement of the user; specifically walking tempo (steps per minute, SPM). D-Jogger matches the tempo of the music (beats per minute, BPM) with the walking tempo of the user, switching songs when appropriate. The main hypothesis is that users will synchronize with the music by aligning their steps to the beats – a form of entrainment.

The idea of such a system is not novel in itself. Yamaha was the first to introduce BODiBeat[1] (2007), followed by Philips Activa[2] (2010). Both devices are focused on selecting music to optimize workout performance, depending on the user's pace and heart rate.

Several other systems have been proposed to do more or less the same thing: [7] and [5] use accelerometers to determine the user's pace, choosing a song from a preprocessed BPM-tagged music library. [7] has the ability to slow down or speed up the music without transposing the pitch. These papers focus on the technical aspect of the systems. [1] hints at the impact of such a system on the user, but the topic is not elaborate. However, none of these applications have been designed or used for research into entrainment, while this is the primary goal of D-Jogger.

2. D-JOGGER

D-Jogger is a framework employed to create a context-sensitive music player, using the user's pace to dynamically choose and adapt the music in real-time. The goal of D-Jogger is to be able to perform research into the phenomenon of entrainment; therefore the system must be flexible and easily adaptable.

Given this prerequisite, we opted to use Max/MSP in designing the application. Max/MSP is a graphical programming environment for real time audio processing that uses objects as basic building blocks, connectable via virtual wires. Objects developed for Max/MSP by third parties are called externals. D-Jogger consists of several externals, connectable in different ways to create a highly flexible framework for the rapid development of applications involving movement analysis and dynamic playlist generation. We give a short description of the available externals and their functioning; Figure 1 provides an overview.

[1] http://www.yamaha.com/bodibeat/
[2] http://www.consumer.philips.com/c/workout-monitors/176850/cat/us/

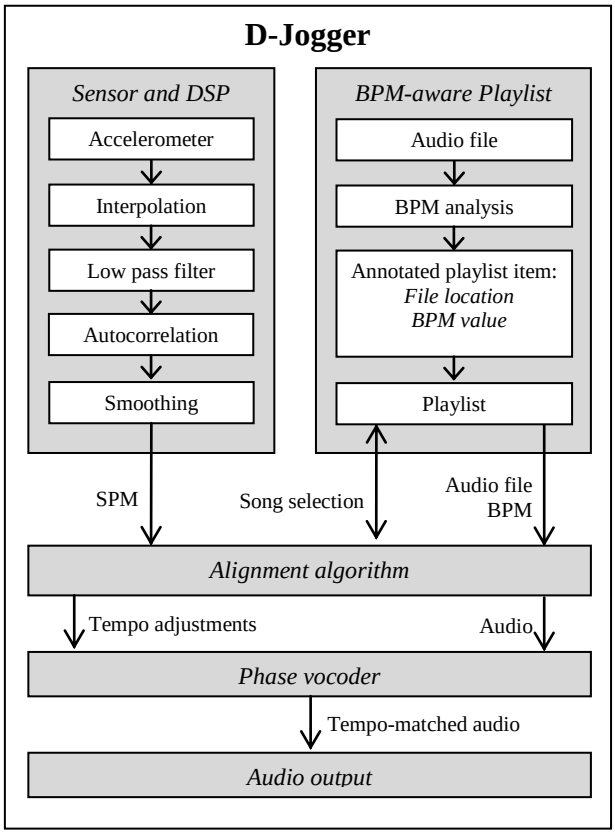


Figure 1. Overview of D-Jogger components

2.1. Sensor and Step Detection

A digital signal processing (DSP) external is used to extract the user's pace from a 3-axis ADXL330 accelerometer signal.

Accelerometer signals vary wildly depending on the gait pattern of the user, location of the sensor, the sensor specifications and the type of surface the user is walking on [8,9]. For example, soft grass, asphalt or treadmills result in different signals. Figure 2 illustrates the signal from different users walking on a treadmill with the sensor attached to their left ankle. The signal is from the axis perpendicular to the ground, showing clearly the moment of impact of the left heel on the treadmill. It is clear that both users' signals are periodic, but differ in other aspects. Several approaches are possible to determine step frequency, such as a Fourier transform or autocorrelation. A comparison of basic step detection algorithms for accelerometers can be found in [12]. For our purposes, we opted to use an autocorrelation algorithm.

The step detection algorithm consists of 4 phases:

- We interpolate the signal at a sample rate of 200 Hz with a second-order polynomial function;
- The resulting signal is filtered using a Finite Impulse Response (FIR) low pass filter of the 10th grade to reduce high frequency noise;
- We apply autocorrelation to the last four seconds of data, which contain at least 2 periods of the step signal. We then look for frequencies in the range of 30 to 125 SPM. Because the sensor is located on only one foot, the lower boundary of 30 SPM is equal to 60 SPM when taking both feet into account.
- The highest lag value is transformed to the SPM value. We take previous SPM values into account to reduce outlier effects, smoothing the results.

The resulting algorithm works independently from the selected axis of the sensor; however results are more stable if an axis with a clear periodic signal is selected.

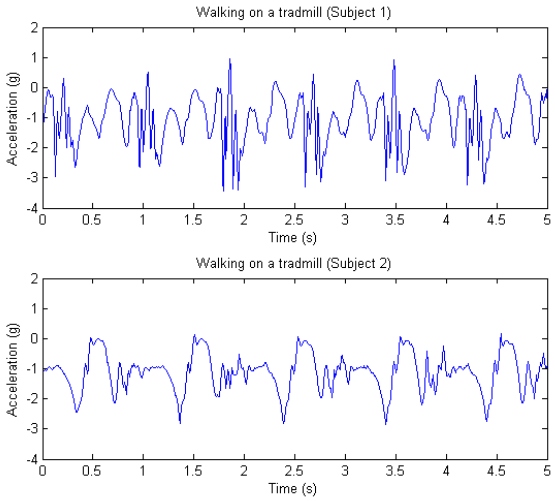


Figure 2. Acceleration signals of the left ankle of 2 subjects on a treadmill

2.2. BPM-Aware Playlist

The playlist external is used to dynamically add or remove songs from the playlist. When adding a song, it is first processed using BeatRoot [3] to extract the tempo information of the song. This includes timestamps of the beats and the BPM value. Optional metadata includes user ratings and general statistics on the usage of the song, preventing frequent repetitions of the same music.

2.3. Phase Vocoder

We use an external to time-stretch the audio without changing the pitch. Several solutions exist for this purpose [4]. ElasticX[3] is a Max/MSP object that allows changing the playback speed and pitch independently. Quality of the resulting audio is very good when the speed remains between 90% and 110% of the original. While this component is not part of our framework itself, it is used intensively and thus listed here.

2.4. Alignment algorithm

A central component connects all available externals. The alignment algorithm acts as the 'moderator' of the music: it decides whether the tempo of the current song should be adapted and whether a new song should be chosen. We present the Dynamic Song and Tempo (DSaT) algorithm, but many other alignment algorithms are possible.

[3] http://www.elasticmax.co.uk

DSaT consists of 2 phases:

- Song Selection: DSaT starts by choosing a song with a BPM close to the SPM value. If no suitable song is found, the SPM value is doubled and the search restarted. This feature is used with very low SPM values (SPM < 90), because very few songs with a BPM less than 90 are available [6].
- Dynamic Tempo: the required tempo adjustment for the song is calculated by dividing the SPM value by the BPM value. When this adjustment falls outside predetermined boundaries for more than 5 seconds, a new song is selected.

3. REPRESENTATIONS OF ENTRAINMENT

D-Jogger is used for research investigating human entrainment to music. We present several ways to visualize this entrainment, each having unique advantages and disadvantages. The following examples are visualizations of the exact timestamps of beats and footsteps from the pilot experiment. We first provide explanations of our chosen methods of visual representation. Results of the pilot experiment will be discussed in the next chapter.

3.1. BPM-SPM Plots

We create a plot where the BPM and SPM values are plotted over time. While this shows very little information about the entrainment itself, it is useful for validating the correct functioning of D-Jogger. We introduce the following notation: b_i is the timestamp of the i^{th} beat in milliseconds, s_j is the timestamp of the j^{th} step in milliseconds, bpm_i is the BPM value between b_{i-1} and b_i and spm_j is the SPM value between s_{j-1} and s_j (analogue for SPM calculation).

$$bpm_i = \frac{1000*60}{(b_i - b_{i-1})} \quad (1)$$

Figure 3 (top) gives an example of such a plot. Over a time span of 130 seconds, four different songs are played, as indicated by the red markers.

3.2. Synchronization plots

A synchronization plot shows the time between each step s_i and the closest beat b_j or b_{j+1}. We note b_j as the beat before s_i, b_{j+1} is the beat after s_i. The synchronization value, $synch_i$, is expressed in milliseconds: 0 means that the beat and the step occurred simultaneously; a positive value means the step occurred before the beat. A higher absolute value of $synch_i$ means a less optimal synchronization.

$$synch_i = \begin{cases} b_j - s_i & \text{if } s_i - b_j \leq b_{j+1} - s_i \\ b_{j+1} - s_i & \text{otherwise} \end{cases} \quad (2)$$

When $synch_i$ equals or is close to 0, we speak of the footsteps and musical beats being *in phase*. When the steps occur exactly midway between two sequential

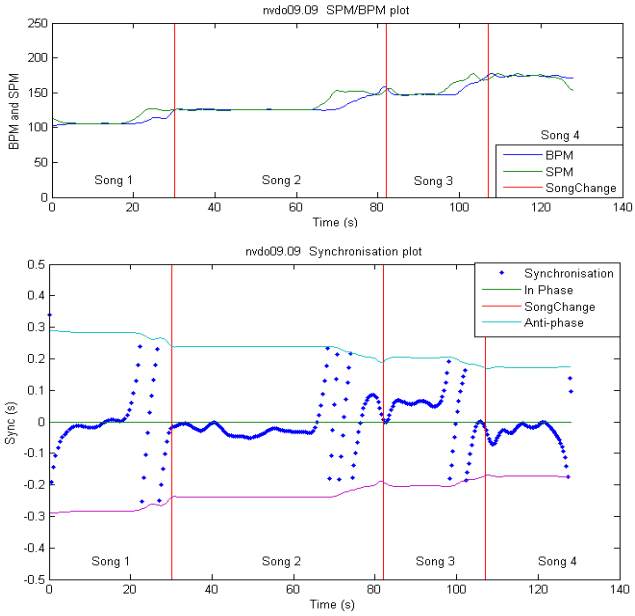


Figure 3. (top) BPM-SPM plot (bottom) Synchronization plot

beats, we speak of them being in *anti-phase*. The anti-phase time is inversely proportional to the BPM. For a visual representation, we plot anti-phase as well. The $synch_i$ value wraps between the positive and negative anti-phase.

Figure 3 (bottom) gives an example of a synchronization plot, representing the same data as the above BPM-SPM plot. This representation is very well suited to provide an overview of the users' actions. We can clearly see when the user is or is close to being in phase (e.g. between 5s and 20s in the graph). When the user speeds up it is clearly visible by the warping of the *synch* value (e.g. 20s to 30s). We can also see the inverse proportional relation of the anti-phase with the BPM value by comparing it with the top plot.

A disadvantage of the synchronization plots is that they are hard to compare to one another. This is due to the variables BPM and song changes. Therefore, we introduce a normalized representation called phase plots.

3.3. Phase plots

Based on a similar approach to that described in [10], we transform $synch_i$ to the phase angle θ_i by dividing by the BPM timing:

$$\theta_i = 2 * \pi * \frac{synch_i}{b_{j+1} - b_j} \quad (3)$$

In the visual representation, a circle is divided into 4 circular sectors. Details about the different sectors can be found in Table 1. The radius represents time, with $r = 1$ being the start of a song and $r = 0.3$ the end of a song. Time is spaced linearly between $r = 1$ and $r = 0.3$.

Using the same data as in the previous examples, we plot the third and forth song on a phase plot (Figure 4). In the figure we can see what we define as in phase and which steps fall within certain sectors. In the synchronization plot, walking appeared to be well synchronized to

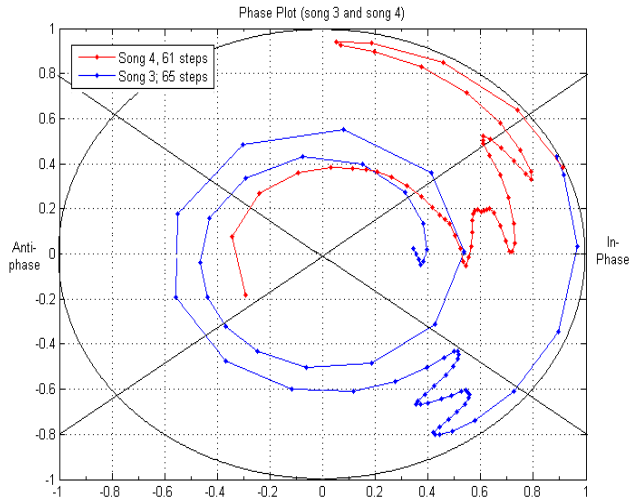


Figure 4. Phase plots of songs 3 and 4

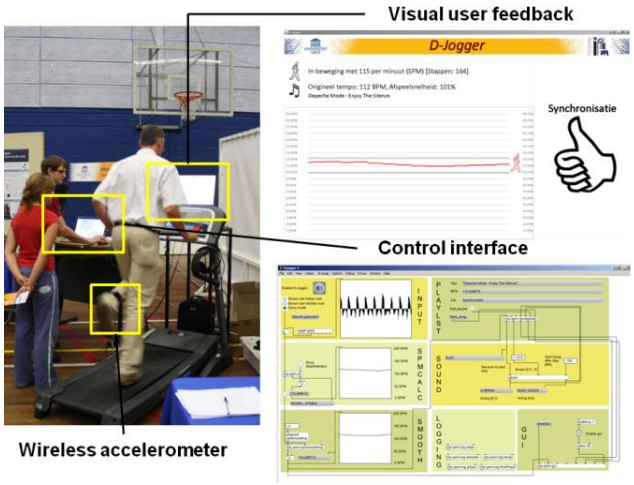


Figure 5. Pilot setup of D-Jogger

the beats of both songs. The phase plots reveal a different picture: the majority of the time, song 3 is not in phase, while the majority of song 4 is in phase. This is confirmed if we look at Table 1: the amount of time spent in a given sector is expressed in the last two columns. We can say that the user synchronized better in the fourth song than in the third. Phase plots and tables make it easier to compare different sets of synchronization data both visually and numerically because of the BPM normalization.

Circular sector	Angles	Song 3	Song 4
In-Phase	0°:39° & 320°:359°	20.31%	66.67%
Anti-phase	140°:219°	7.81%	3.33%
Rest	40°:139° & 220°:319°	71.86%	30.00%

Table 1. Definition of circular sectors on phase plots and two examples of time spend in each sector for songs 3 and 4

3.4. Phase histogram plots

Next, we create a histogram of the phase angles. A phase histogram gives a global overview of the amount of time a user spends in a certain phase angle, giving an overview of the user's entrainment. Figure 6 illustrates the phase histogram for all available data from the pilot experiment. The advantages of this representation are clear: independent of the number of steps and easily comparable with different histograms, they give a clear overview of entrainment. These results can also be represented in a table for easier statistical analysis.

4. PILOT EXPERIMENT

4.1. Setup

We designed a version of D-Jogger for demonstration purposes, using a 3-axis ADXL330 accelerometer attached at the user's ankle and connected via Bluetooth. A graphical user interface (GUI) was placed in front of the user, showing the user's SPM history, music information and the synchronization of BPM and SPM. The GUI did not include information about the entrainment itself. Music was chosen from a library containing 50 pop songs. The DSaT algorithm was used for music selection and real-time adaptation. If the necessary tempo adjustment was outside the boundaries of ±10% for more than 5 seconds, a new song was chosen with a BPM closer to the current SPM. These boundaries were also shown in the GUI. After using D-Jogger, participants were asked to complete a survey. Figure 5 illustrates the experimental setup.

4.2. Participants, experiment and survey

A total of 33 participants experimented with the system at a public event in 2009. The participants were given a short introduction prior to the experiment, explaining the concept of time-stretching music to the users' pace. Participants were not informed of the concept entrainment.

After the introduction, the sensor was attached to their left ankle. Once on the treadmill, users were free to experiment with D-Jogger, picking and changing their speed without time limitations. The treadmill control board featured speed presets (2, 4 … 20 km/h) and fine speed control (0.1 km/h).

After the experiment, participants filled in a survey asking their personal jogging preferences and their experience with D-Jogger.

4.3. Data and analysis

A Zoom H4 stereo recorder was used to record the whole experiment. One channel recorded the audio as perceived by the user; the other channel recorded the footsteps using a microphone placed near the front of the treadmill. This was done to have a dataset independent of the D-Jogger logs in order to verify the correct functioning of D-Jogger itself. From the recordings we manually extracted the timestamps of beats and steps. A total of 15 datasets were usable, giving a total of 3700 steps. Various parts of the recording were unusable because of unclear step recordings.

4.4. Results

We calculated the synchronization phase for 3700 steps. Figure 6 depicts the results in a phase histogram. This shows that the majority of steps (56.79 %) were in sync with the music. Because D-Jogger does not do phase synchronization, the user was responsible for the entrainment effect between the music and their gait pattern. 17.37% steps were taken in anti-phase, the remaining 25.85% of steps were neither in phase or in anti-phase and mostly occurred close to song changes.

Our preliminary conclusions are that:

- While using D-Jogger, the majority of subjects synchronized to the beats, no matter what the users' pace.
- If the music tempo is close enough to the user's pace, the user tends to synchronize his or her steps with the beats.
- Users reported in the survey that they feel more motivated when in sync with the music.

5. DISCUSSION

The question arises as to whether a treadmill stimulates or hinders entrainment. One might say that the SPM of the user is dependent on the speed selected on the treadmill. While this is partially true, a user can speed up or slow down the SPM value significantly by adjusting the step length, as seen in several datasets. This enables the user to synchronize with the music on a treadmill. However, we do not know to what extent the use of a treadmill influences the entrainment process. Future studies will include a comparison of uninhibited locomotion activity and locomotion activity on a treadmill.

While this was as small study, some interesting results were procured. They do require confirmation by new experiments in more rigorous settings, but the pilot experiment gives us hints about the expected results. The result of the survey, where users reported feeling more motivated when in sync with the music, is compelling but also subjective and requires further validation.

Currently, the prototype of D-Jogger has only been tested using songs with a typical 4/4 time signature and constant beat. This is partly due to the limitations of the beat extraction, which works optimal with the simple and more popular Western songs. With the goal of D-Jogger in mind, we require songs with an unambiguous and clear rhythm, so in this early stage we do not see this as a problem.

Another important remark is that when a new song is chosen, phase synchronization could be temporarily lost. Currently, a new song starts independently of the time when footsteps occur, so there is a high chance that the user will be out of sync with the music immediately following a song change. For this we intend to analyze the sensors for phase information of the steps so a song can be started in accordance with the step phase.

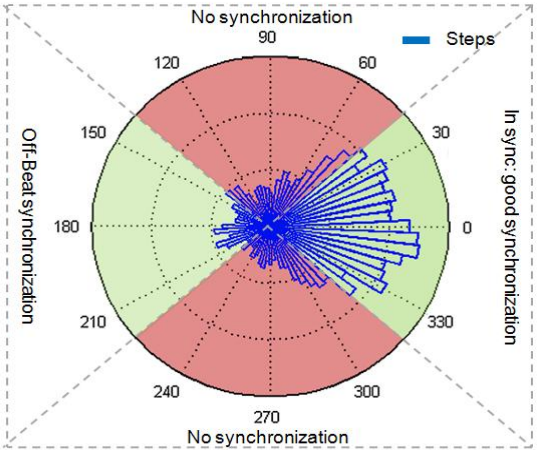


Figure 6. Results of the pilot experiment

6. FUTURE WORK

The most limiting factor up until now has been the use of a stationary version of D-Jogger with a treadmill. A mobile version will be implemented, using the internal accelerometer of the mobile device. With the mobile version we can perform an experiment with unconstrained locomotion, yielding results closer to our typical real-life experience with walking to music.

The mobile version will also feature a true multimodal interface: user control will be possible using simple gestures on a touch screen. This multimodal interface is ideal for use in training sessions because the user can select and rate music without interrupting the training.

The D-Jogger framework can also be used in other applications. We are currently looking into rehabilitation therapy for Parkinson patients using audio cues and a personal training device.

7. CONCLUSIONS

While there is still much to be done, D-Jogger shows great potential in both the current application and future possibilities involving entrainment research in body movement in relation to music.

We proposed several options to visualize entrainment data. BPM-SPM plots show the evolution of tempo of both the music and the user in time, while the complementary synchronization plots show the evolution of entrainment in time. Both are very useful for visual inspection, while phase tables, plots and histograms are better suited for analytical purposes. They show the synchronization data in a normalized way, allowing comparison of different datasets.

We performed a pilot study using the D-Jogger. The results show our hypothesis, that entrainment occurs when SPM is close to BPM, to be correct. However, further research is necessary using both the stationary and mobile version of D-Jogger to confirm the hypothesis.

REFERENCES

[1] J.T. Biehl, P.D. Adamczyk and B.P. Bailey, "DJogger: a mobile dynamic music device". *In CHI '06 Extended Abstracts on Human Factors in Computing Systems* pp. 556-561, 2006

[2] M. Clayton, R. Sager and U. Will, "In Time with the Music: The Concept of Entrainment and its Significance for Ethnomusicology", *European Meetings in Ethnomusicology*, Vol. 11, pp. 3–142, 2005.

[3] S. Dixon, "Evaluation of the Audio Beat Tracking System BeatRoot", *Journal of New Music Research*, Vol. 36, pp. 39-50, 2007

[4] D. Dorran, B. Lawlor and E. Coyle, "A comparison of time-domain timescale modification algorithms", *In 120th Audio Engineering Society Convention*, issue paper 6674, 2006.

[5] G.T. Elliott, B. Tomlinson, "PersonalSoundtrack: context-aware playlists that adapt to user pace", *CHI '06 extended abstracts on Human factors in computing systems*, 2006.

[6] F. Gouyon, "Dance music classification: A tempo-based approach", *In Proceedings of the International Conference on Music Information Retrieval*, Vol. 2004, pp. 501-504, 2004.

[7] J. Hockman, M.M. Wanderley and I. Fujinaga, "Phase vocoder manipulation by runner's pace", *In Proceedings of the 2009 Conference on New Instruments for Musical Expression*, pp. 90-93, 2009.

[8] L. Hongyin and T. Kaiyu, "The reliability of using accelerometer and gyroscope for gait event identification on persons with dropped foot", *Gait & Posture*, Vol. 27, Issue 2, pp. 248-257, 2008.

[9] H.-J Jang, J.W. Kim and D.-H. Hwang, "Robust step detection method for pedestrian navigation systems", *Electronics Letters*, vol. 43, no. 14, 2007.

[10] J.A.S. Kelso, *Dynamic Patterns: The Self-Organization of Brain and Behavior*. Cambridge, MA: The MIT Press, 1995.

[11] H. G. MacDougall and S. T. Moore, "Marching to the beat of the same drummer: the spontaneous tempo of human locomotion", *Journal of Applied Physiology,* vol. 99, pp. 1164-1173, 2005.

[12] M. Marschollek, M. Goevercin, K.H. Wolf, B. Song, M. Gietzelt, R. Haux and E. Steinhagen-Thiessen, "A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons". *Conference Proceedings IEEE Eng Med Biol Soc 2008*, pp. 1319-1322, 2008.

[13] F. Styns, L.V. Noorden, D. Moelants and M. Leman, "Walking on music", *in Human Movement Science*, Vol. 26, Issue 5, pp. 769-785, 2007.

LEGATO AND GLISSANDO IDENTIFICATION IN CLASSICAL GUITAR

Tan Hakan Özaslan and Josep Lluis Arcos
Artificial Intelligence Research Institute, IIIA.
Spanish National Research Council, CSIC.
{tan,arcos}@iiia.csic.es

ABSTRACT

Understanding the gap between a musical score and a real performance of that score is still a challenging problem. To tackle this broad problem, researchers focus on specific instruments and/or musical styles. Hence, our research is focused on the study of classical guitar and aims at designing a system able to model the use of the expressive resources of that instrument. Thus, one of the first goals of our research is to provide a tool able to automatically identify expressive resources in the context of real recordings. In this paper we present some preliminary results on the identification of two classical guitar articulations from a collection of chromatic exercises recorded by a professional guitarist. Specifically, our system combines several state of the art analysis algorithms to distinguish among two similar guitarists' left hand articulations such as legato and glissando. We report some experiments and analyze the results achieved with our approach.

1. INTRODUCTION

An affective communication between listeners and performers can be achieved by the use of instruments' expressive resources [1, 2, 3]. Expressive resources play also an important role to clarify the musical structure of a piece [4, 5, 6]. Although each instrument provides a collection of specific expressive capabilities, its use may vary depending on the musical genre or the performer.

Our research on musical expressivity is focused on the study of classical guitar and aims at designing a system able to model the use of the expressive resources of that instrument. As a first stage of our research we are developing a tool able to automatically identify the use of guitar articulations.

There are several studies on plucked instruments and guitar synthesis such as on extraction of expressive parameters for synthesis [7, 8]. However, expressive articulation analysis from real guitar recordings has not been fully tackled. The analysis of a guitar performance is complex because guitar is an instrument with a rich repertoire of expressive articulations.

In guitar playing both hands are used: one hand is used to press the strings in the fretboard (commonly the left hand) and the other to pluck the strings. Strings can be plucked using a single plectrum called a flatpick or by directly using the tips of the fingers. The hand that presses the frets is mainly determining the notes while the hand that plucks the strings is mainly determining the note onsets and timbral properties. However, left hand is also involved in the creation of a note onset and different expressive articulations like legato, glissando, grace notes, or vibratos [8].

In a previous research [10], we proposed a system able to detect attack-based articulations and distinguish among legato and grace notes. The goal of this paper is to extend the capabilities of the existing system to distinguish among legato and glissando articulations. In both, legato and glissando, left hand is involved in creation of the note onset.

In the case of ascending legato, after plucking the string with the right hand, one of the fingers of the left hand (not already used for pressing one of the frets), presses a fret causing another note onset. Descending legato is performed by plucking the string with a left-hand finger that was previously used to play a note (i.e. pressing a fret).

The case of glissando is similar but this time after plucking one of the strings with the right hand, the left hand finger that is pressing the string is slipped to another fret also generating another note onset. Notice that we are not considering here grace notes that are played in a similar way than glissando.

When playing legato or glissando on guitar, it is common for the performer to play more notes within a beat than the stated timing enriching the music that is played. A powerful legato and glissando can be differentiated between each other easily by ear. However, in a musical phrase context where the legato and glissando are not isolated, it is hard to differentiate among these two expressive articulations.

The structure of the paper is as follows: Section 2 describes our methodology for legato and glissando determination and differentiation. Specifically, our approach uses aperiodicity information to identify articulations, histograms to compute the density of the peak locations, and a symbolic aggregate approximation (SAX) representation to characterize the articulation models. Next, Section 3 focuses on the experiments conducted to evaluate our system. Last section, Section 4, summarizes current results and proposes the next research steps.

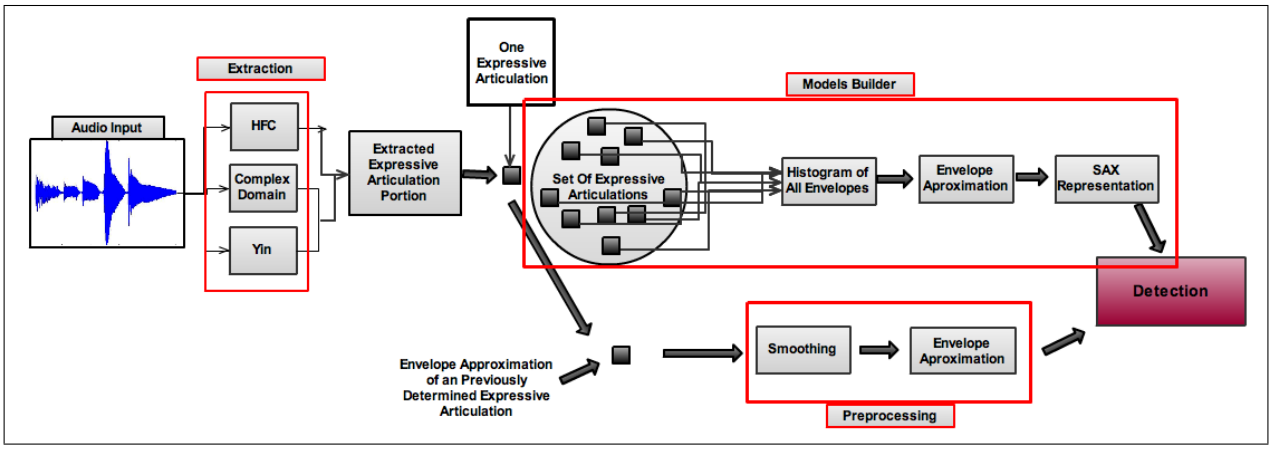


Figure 1: Main diagram of our model, which contains three sub models; Extraction, Model Builder and Preprocessing

2. SYSTEM ARCHITECTURE

In this paper we propose a new system able to identify two expressive articulations: legato and glissando. To that purpose we use a *Region Extraction* module that is part of a previous development [10]. The regions identified by the region extraction module are the inputs to the new components: the *Models Builder* and the *Detection* component (see Figure 1). In this section, first we briefly present the region *Extraction*. Next, we describe our preliminary research to select the appropriate descriptor to analyze the behavior of legato and glissando. Finally, we explain the new two components, Model Builder and Detection.

2.1 Extraction of Candidates

Guitar performers can apply different articulations by using both of their hands. However, the kind of articulations that we are investigating (legato and glissando) are performed by the left hand. Although they (legato and glissando) can cause onsets, these onsets are not as powerful in terms of energy and also have different characteristics in terms of harmonicity, comparing to the plucking onsets [11]. Therefore, we need an onset determination algorithm suitable to differentiate between plucking onsets and left-hand onsets.

The first task of the extraction module is to determine the onsets caused by the plucking hand, i.e. right hand onsets. As right hand onsets are more percussive than left hand onsets we use a measure appropriate to this feature. HFC is a measure taken across a signal spectrum and can be used to characterize the amount of high-frequency content (HFC) in the signal [12]. As Brossier [13] stated, High Frequency Content (HFC) measure is effective with percussive onsets but less successful determining non-percussive or legato phrases. Then, HFC is sensitive for abrupt onsets but not enough sensitive to the changes of fundamental frequency caused by the left hand.

Aubioonset library [14] gave us the opportunity to tune the peak-picking and silence threshold. One of the key stages of candidate extraction is to optimize peak-picking and silence thresholds in a way that only the plucking hand onsets are determined and the pitch changes due to legato

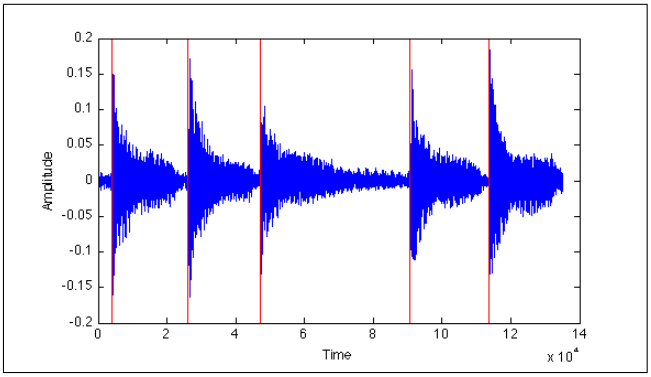


Figure 2: High Frequency Content onsets from the Region Extraction module.

or glissando are not determined as onsets. In order to find suitable parameters for this goal, before running our model, we used a set of hand annotated recordings. Our set is a concatenated audio file which contains 24 non-expressive notes, 6 glissando and 6 legato notes. We hand annotated the onsets of non-expressive, legato and glissando notes. What we want to obtain from Aubioonset was the onsest of the plucking hand. Since this annotated set contains the exact places of onset that we want to obtain from Aubioonset, this set can be considered as our ground truth. After testing with different parameters, we achieved the best results with the following values for algorithm parameters: 1 for peak-picking threshold and $-85db$ for silence threshold.

An example of the resulting onsets proposed by HFC is shown in Figure 2. Specifically, in the exemplified recording six notes are played following the pattern detailed in experiments (see Figure 16) where only 5 of them are plucking onsets. In Figure 2 detected onsets are marked as vertical lines. Between third and fourth detected onsets an expressive articulation (legato) is present. Thus, HFC succeeds because it only determines the onsets caused by the right hand.

The second task performed by the extraction module is to analyze the sound fragment between two onsets. First, each portion between two plucking onsets is analyzed individually. Specifically, two points are determined: the *end*

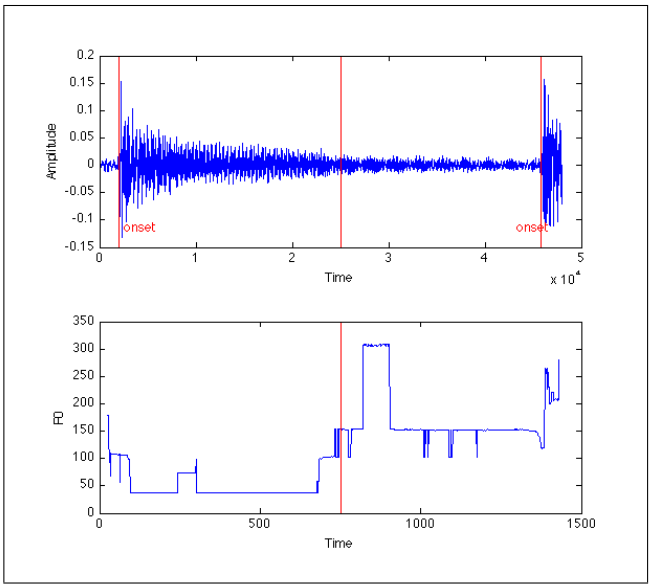


Figure 3: Example of detection of a candidate to an expressive articulation.

of the attack and the *release start*. From experimental measures, we determined attack finish position as $10ms$ after the amplitude reaches its local maximum. We determined the release start position as the final point where local amplitude is equal or greater than 3 percent of the local maximum. Only the fragment between these two points is considered for the further analysis because the noisiest part of a signal is the attack part and the release part of a signal contains unnecessary information for pitch detection (see [15] for details).

We use additional algorithms with a lower threshold in order to capture the changes in fundamental frequency inside the sound fragment. Specifically, complex domain algorithm [16] is used to determine the peaks and Yin [17] is used for the fundamental frequency estimation. Figure 3 shows fundamental frequency evolution between the central region presented in Figure 2. The change of frequency detected points out a possible candidate of expressive articulation. More details can be found in [10].

2.2 Selecting a Descriptor

After extracting the regions candidates to contain expressive articulations, the next step was to analyze them. Because different expressive articulations (legato vs glissando) should present different characteristics in terms of changes in amplitude, aperiodicity, or pitch[8], we focused the analysis on comparing these deviations.

We built representations of these three features (amplitude, aperiodicity, and pitch). Representations helped us to compare different data with different length and density. As we stated above, we are mostly interested in changes: changes in High Frequency Content, changes in fundamental frequency, changes in amplitude, etc. Therefore, we explored the peaks in the examined data because peaks are the points where changes occur.

As an example, Figures 4 and 5 show, from top to bottom, amplitude evolution, pitch evolution, and changes in aperiodicity. As both Figures show, *glissando* and *legato* examples, the changes in pitch are similar. However, the changes in amplitude and aperiodicity present a characteristic slope.

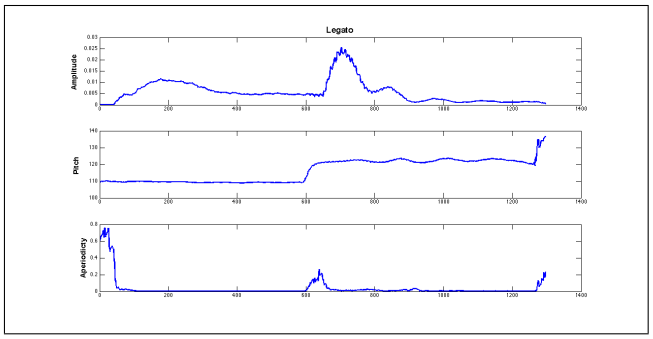


Figure 4: Different features of a legato example. From top the bottom, representations of amplitude, pitch and aperiodicty of the examined legato region.

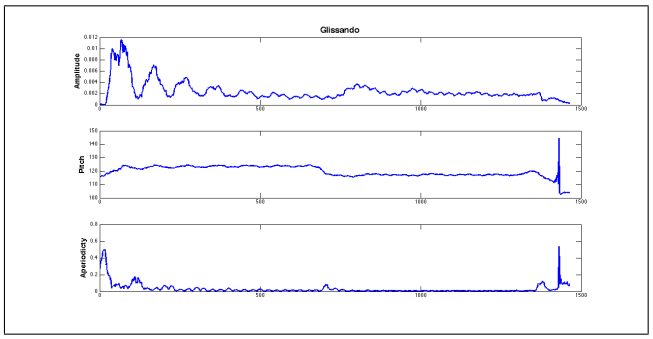


Figure 5: Different features of a glissando example. From top the bottom, representations of amplitude, pitch and aperiodicty of the examined Glissando region.

So, as a first step we concentrated on determining which descriptor could be used. To make this decision, we built models for both aperiodicty and amplitude by using a set of training data. The details of this model construction will be explained in Section 2.4. As a result, we obtained two models (for amplitude and aperiodicity) for both legato and glissando as is shown in Figure 6 and Figure 7. Analyzing the results, amplitude is not a good candidate because the models behave similarly. In contrast, aperiodicity models present a different behavior. Therefore, we selected aperiodicity as the descriptor.

2.3 Preprocessing

Before analyzing and testing our recordings, we applied two different preprocessing techniques to the data in order to make them smoother and ready for comparison.

2.3.1 Smoothing

As expected, aperiodicity portion of the audio file that we are examining includes noise. Our first concern was to avoid this noise and obtain a nicer representation. In order to do that first we applied a 50 step running median smoothing. Running median smoothing is also known as

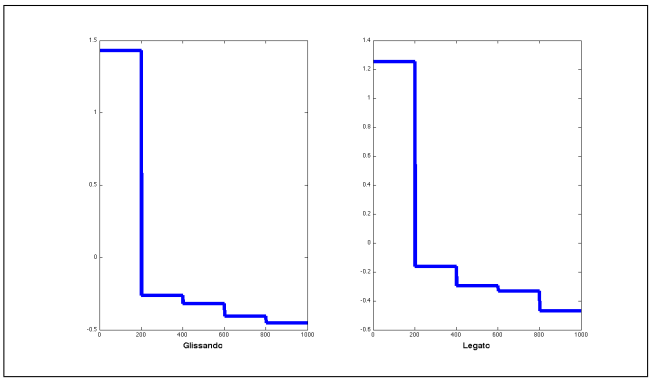


Figure 6: Amplitude models of glissando and legato.

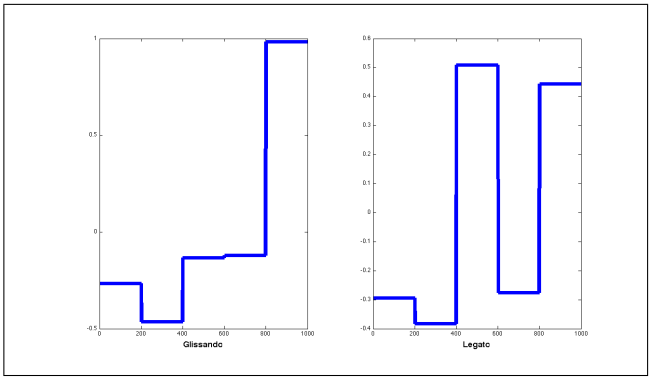


Figure 7: Aperiodicity models of glissando and legato.

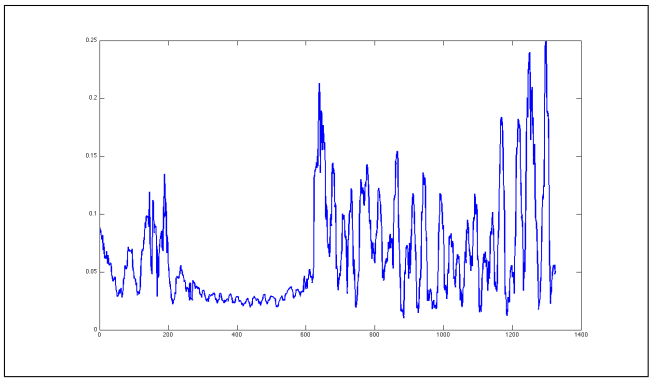


Figure 8: Aperiodicity .

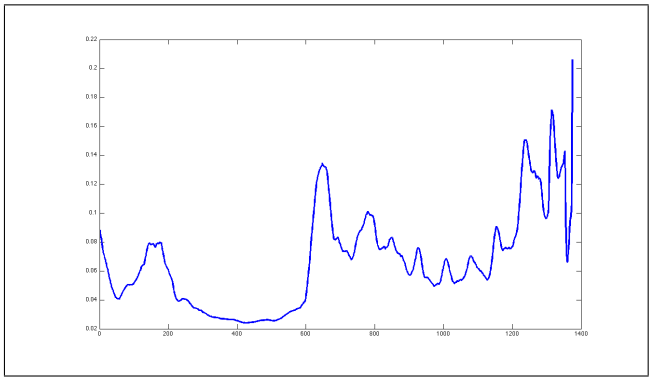


Figure 9: Smoothed Aperiodicity.

median filtering. Median filtering is widely used in digital image processing because under certain conditions, it preserves edges whilst removing noise. In our situation since we are interested in the edges and in removing noise, this approach fits our purposes. By smoothing, the peaks locations of the aperiodicity curves become more easy to extract. Figure 8 and Figure 9 exemplify the smoothing process and show the results we pursued.

2.3.2 Envelope Approximation

After obtaining a smoother data, an envelope approximation algorithm was applied. The core idea of the envelope approximation is to obtain a fixed length representation of the data, specially considering the peaks and also avoiding small deviations by connecting these peak approximations linearly. The envelope approximation algorithm has three parts: *peak peaking*, *scaling* of peak positions according to a fixed length, and *linearly connecting* the peaks. After the envelope approximation, all the data regions we are investigating had the same length, i.e. regions were compressed or enlarged depending their initial size.

We collect all the peaks above a pre-determined threshold. Next, we scale all these peak positions. For instance, imagine that our data includes 10000 bins and we want to scale this data to 1000. And lets say, our peak positions are : 1460, 1465, 1470, 1500 and 1501. What our algorithm does is to scale these peak locations dividing all peak locations by 10 (since we want to scale 10000 to 1000) and round them. So they become 146, 146, 147, 150 and 150. As seen, we have 2 peaks in 146 and 150. In order to fix this duplicity, we choose the ones with the highest peak. After collecting and scaling peak positions, the peaks are linearly connected. As shown in Figure 10, the obtained graph is an approximation of the graph shown in Figure 9. Linear approximation helps the system to avoid consecutive small tips and dips.

In our case all the recordings were performed at 60bpm and all the notes in the recordings are 8th notes. That is, each note is half a second, and each legato or glissando portion is 1 second. We recorded with a sampling rate of 44100, and we did our analysis by using a hop size of 32 bins, i.e. 44100/32 = 1378 bins. We knew that this was our highest limit. For the sake of simplicity, we scaled our x-axis to 1000 bins.

2.4 Building the Models

After applying the preprocessing techniques, we obtained equal length aperiodicity representations of all our expressive articulation portions. Next step was to construct models for both legato and glissando by using this data. In this section we describe how we constructed the models cited briefly in the Section 2.2 (and shown in Figure 6 and Figure 7). The following steps were used to construct the models: *Histogram Calculation*, *Smoothing* and *Envelope approximation* (explained in Section 2.3), and finally, SAX representation. In this section we present the Histogram Calculation and the SAX representation techniques.

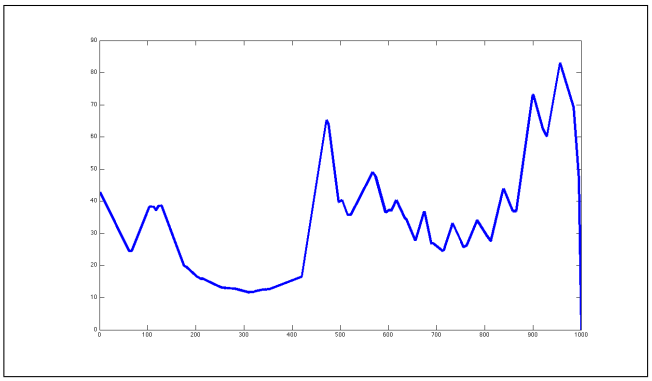
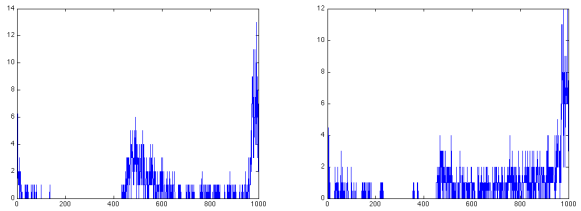


Figure 10: Envelope approximation of a legato portion.



(a) Legato Histogram (b) Glisando Histogram

Figure 11: Peak histograms of our legato and glissando training sets.

2.4.1 Histogram Calculation

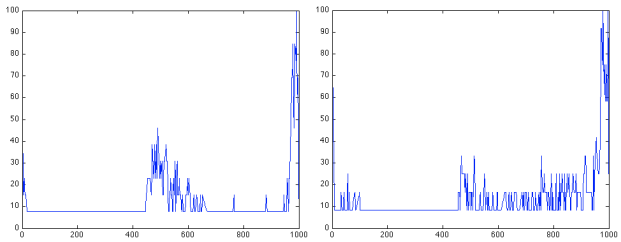
Another method that we are using is histogram envelope calculation. We use this technique to calculate the peak density of a set of data. Specifically, a set of recordings containing 36 legato and 36 glissando examples (recorded by a professional classical guitarist) was used as training set. First, for each legato and glissando example, we determined the peaks. Since we want to model the places where condensed peaks occur, this time we use a threshold which is 30 percent and collect the peaks which have amplitude values above this threshold. Notice that the threshold is different than the used in envelope approximation. Then, we used histograms to compute the density of the peak locations. Figure 11 shows the resulting histograms.

After constructing the histograms, as shown in Figure 11, we used our envelope approximation method to construct the envelopes of legato and glissando histogram models (see Figure 12).

2.4.2 SAX: Symbolic Aggregate Approximation

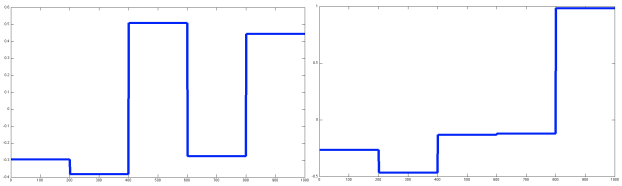
Although the histogram envelope approximations of legato and glissando in Figure 12 are close to our purposes, they still include noisy sections. Rather than these abrupt changes (noises), we are interested in a more general representation reflecting the changes more smoothly.

SAX (Symbolic Aggregate Approximation) [18], is a symbolic representation used in time series analysis that provides a dimensionality reduction while preserving the properties of the curves. Moreover, SAX representation makes the distance measurements easier. Then, we applied



(a) Legato Final Envelope (b) Glisando Final Envelope

Figure 12: Final envelope approximation of peak histograms of legato and glissando training sets.



(a) Legato SAX Representation (b) Glisando SAX Representation

Figure 13: SAX representation of legato and glissando final models.

the SAX representation to histogram envelope approximations.

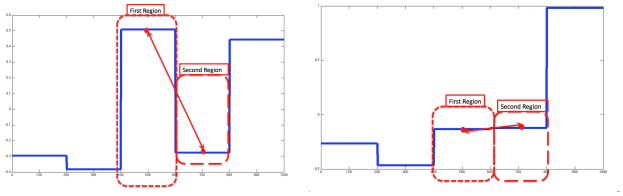
As we mentioned in Section 2.3.2, we scaled the x-axis to 1000. We made tests with step sizes of 10 and 5. As we report in the Experiments section, an step size of 5 gave better results. We also tested with step sizes lower than 5, but the performance clearly decreased. Since we are using an step size of 5, each step becomes 100 bins in length. After obtaining the SAX representation of each expressive articulation, we used our distance calculation algorithm which we are going to explain in the next section.

2.5 Detection

After obtaining the Sax representation of our glissando and legato models, we divided them into 2 regions, a first region between bins 400 and 600, and a second region between bins 600 and 800 (see Figure 14).

For the expressive articulation excerpt, we have the envelope approximation representation with the same length of the SAX representation of final models. So, we can compare the regions. For the final expressive articulation models (see Figure 13) we took the value for each region and compute the deviation (slope) between these two regions. We make this computation for both legato and glissando models separately.

We also compute the same deviation for each expressive articulation envelope approximation (see Figure 15). But this time, since we do not have SAX representation, for each region we do not have single values. Therefore, for each region we compute the local maxima and take the deviation (slope) of these two local maxima. After obtaining this value, we compare it with the numbers that we obtained from both final models of legato and glissando. If



(a) Legato (b) Glisando

Figure 14: Peak occurrence deviation.

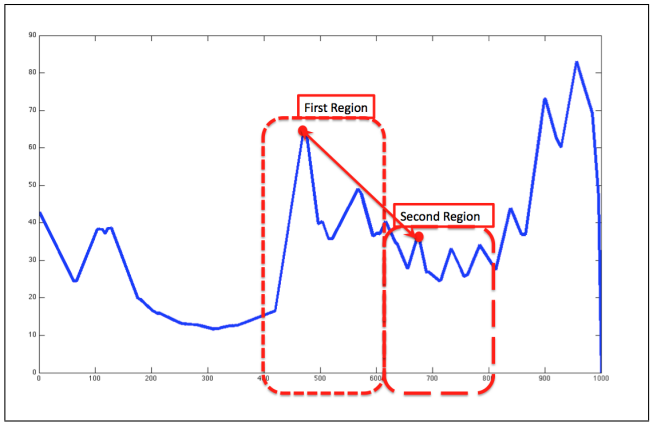


Figure 15: Expressive articultion difference.

the deviation value is closer to the legato model, we annotate this expressive articulation as a legato and vice versa.

3. EXPERIMENTS

The goal of the experiments was to test the accuracy of our approach. Because legato and glissando can be played in ascending or descending intervals, we were also interested in studying the results considering these two movements. Additionally, since in a guitar there are three nylon and three metallic strings, we also studied the results on these two sets of strings.

Borrowing from Carlevaro's guitar exercises [19], we recorded a collection of ascending and descending chromatic scales. Legato and glissando examples were recorded by a professional classical guitar performer. The performer was asked to play chromatic scales in three different regions of the guitar fretboard. Specifically, we recorded notes from the first 12 frets of the fretboard where each recording concentrated in 4 specific frets. The basic exercise from the first fretboard region is shown in Figure 16. Each scale contains 24 ascending and 24 descending notes. Each exercise contains 12 expressive articulations (the ones connected with an arch). Since we repeated the exercise at three different positions, we obtained 36 legato and 36 glissando examples.

We presented all the 72 examples to our system. Then, our system proposed a possible expressive articulation as described in Section 2. Results are reported in Table 1.

First, we may observe that a step size of 5 is the most appropriate setting. This result corroborates that a higher resolution when discretizing is not required and demonstrates

	Step Size	
Recordings	**5**	**10**
Ascending Legato	100%	100%
Descending Legato	66.6%	72.2%
Ascending Glissando	83.3%	61.1%
Descending Glissando	77.7%	77.7%
Glissando Metallic Strings	77.7%	77.7%
Glissando Nylon Strings	83.3%	61.1%
Legato Metallic Strings	86.6%	80%
Legato Nylon Strings	73.3%	86.6%

Table 1: Performance of our model applied to test set

that the SAX representation provides a powerful technique to summarize the information about changes.

The overall performance for legato identification is 83.5%. Notice that ascending legato reaches a 100% of accuracy whereas descending legato achieves a 66.6%. Regarding glissando, there is no difference between ascending or descending accuracy (83.3%,77.7%). Finally, analyzing the results when considering the string type, the results show a similar accuracy.

4. CONCLUSION

In this paper we presented some preliminary results on the identification of two classical guitar articulations, legato and glissando, from a collection of chromatic exercises recorded by a professional guitarist. Our approach uses aperiodicity information to identify the articulation and a SAX representation to characterize the articulation models.

Reported results show that our system is able to differentiate successfully among these two articulations. Our next goal is to study the capabilities of our approach in the context of a real performance. To avoid the analysis on a polyphonic recording, we plan to use an hexaphonic pickup.

5. ACKNOWLEDGMENTS

This work was partially funded by NEXT-CBR (TIN2009-13692-C03-01), IL4LTS (CSIC-200450E557) and by the Generalitat de Catalunya under the grant 2009-SGR-1434. Tan Hakan Özaslan is a Phd student of the Doctoral Program in Information, Communication, and Audiovisuals Technologies of the Universitat Pompeu Fabra.

6. REFERENCES

[1] P. Juslin, "Communicating emotion in music performance: a review and a theoretical framework," in *Music and emotion: theory and research* (P. Juslin and J. Sloboda, eds.), pp. 309–337, New York: Oxford University Press, 2001.

[2] E. Lindström, "5 x "oh, my darling clementine". the influence of expressive intention on music performance," 1992. Department of Psychology, Uppsala University.

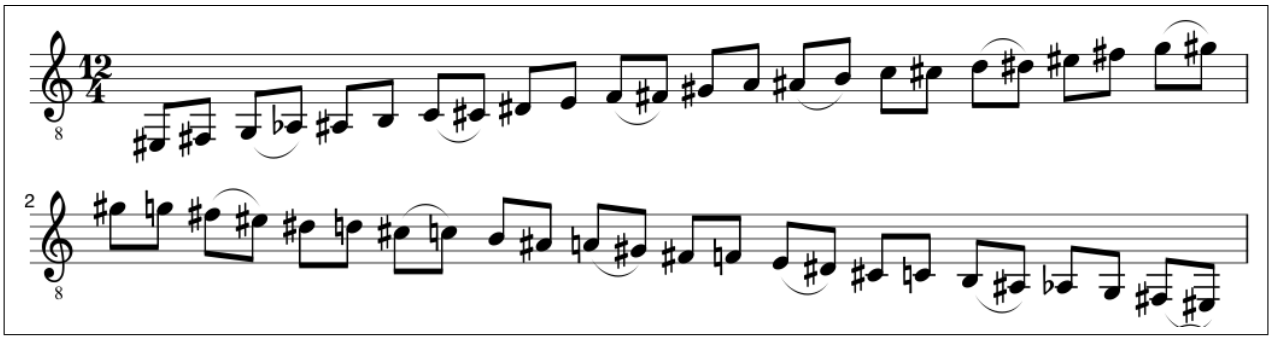


Figure 16: Legato Score in first position.

[3] A. Gabrielsson, "Expressive intention and performance," in *Music and the Mind Machine* (R. Steinberg, ed.), pp. 35–47, Berlin: Springer-Verlag, 1995.

[4] J. A. Sloboda, "The communication of musical metre in piano performance," *Quarterly Journal of Experimental Psychology*, vol. 35A, pp. 377–396, 1983.

[5] A. Gabrielsson, "Once again: The theme from Mozart's piano sonata in A major (K. 331). A comparison of five performances," in *Action and perception in rhythm and music* (A. Gabrielsson, ed.), pp. 81–103, Stockholm: Royal Swedish Academy of Music, 1987.

[6] C. Palmer, "Anatomy of a performance: Sources of musical expression," *Music Perception*, vol. 13, no. 3, pp. 433–453, 1996.

[7] C. Erkut, V. Valimaki, M. Karjalainen, and M. Laurson, "Extraction of physical and expressive parameters for model-based sound synthesis of the classical guitar," in *108th AES Convention*, pp. 19–22, February 2000.

[8] J. Norton, *Motion capture to build a foundation for a computer-controlled instrument by study of classical guitar performance*. PhD thesis, Stanford University, September 2008.

[9] H. Heijink and R. G. J. Meulenbroek, "On the complexity of classical guitar playing:functional adaptations to task constraints," *Journal of Motor Behavior*, vol. 34, no. 4, pp. 339–351, 2002.

[10] T. Ozaslan, E. Guaus, E. Palacios, and J. Arcos, "Attack based articulation analysis of nylon string guitar," in *CMMR 2010*, June 2010.

[11] C. Traube and P. Depalle, "Extraction of the excitation point location on a string using weighted least-square estimation of a comb filter delay," in *In Procs. of the 6th International Conference on Digital Audio Effects (DAFx-03*, 2003.

[12] P. Masri, *Computer modeling of Sound for Transformation and Synthesis of Musical Signal*. PhD thesis, University of Bristol, 1996.

[13] P. Brossier, J. P. Bello, and M. D. Plumbley, "Real-time temporal segmentation of note objects in music signals," in *Proceedings of the International Computer Music Conference (ICMC2004)*, November 2004.

[14] P. Brossier, *Automatic annotation of musical audio for interactive systems*. PhD thesis, Centre for Digital music, Queen Mary University of London, 2006.

[15] C. Dodge and T. A. Jerse, *Computer Music: Synthesis, Composition, and Performance*. Macmillan Library Reference, 1985.

[16] C. Duxbury, J. Bello, J. Davies, S. M., and M. Mark, "Complex domain onset detection for musical signals," in *Proceedings Digital Audio Effects Workshop*, 2003.

[17] A. de Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[18] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, October 2007.

[19] A. Carlevaro, "Serie didactica para guitarra," vol. 4, Barry Editorial, 1974.

TONAL SIGNATURES

Gilbert Nouno
Researcher, Electronic Musician - Ircam
gilbert.nouno@ircam.fr

Malik Mezzadri
Composer, Improviser - Label Bleu
malik.mezzadri@gmail.com

ABSTRACT

We present in this paper an original approach of the use of tonality for composition and improvisation, developed by the French flute player, composer and improviser Malik Mezzadri. The main concept is about finding a minimal group of notes which acts as a *signature* of a given scale in the major-minor tonal system. We will first define the notion of tonal signatures in the tonal system and expose its principles. Among the possible way to solve our problem and find all the *tonal signatures*, we define some constraints and we use a constraint solver implemented in *Open Music*[1], a computer assisted music and composition environment. We provide some examples of original compositions along with improvisation playing based on the tonal signature concept. Malik Mezzadri's music counts already a rich discography with players from the international jazz scene.

1. TONAL SIGNATURES

1.1 In the context of major-minor tonal system

Tonal signatures are subsets of notes singled out from the tonal system. We consider the major-minor tonal system as referred and characterized by the three following main modes (Table 1 and Figure 1) :

Table 1. The main modes of major-minor tonal system

Modes	*Intervalic structure*
Major	$[T, T, H]\ T\ [T, T, H]$
minor	$[T, H, T]\ T\ [T, T, H]$
harmonic minor	$[T, H, T]\ T\ [H, T\&H, H]$

In Table 1. the letter T represents a whole tone interval, the letter H a half tone interval, and $T\&H$ represents one and a half tone interval. The brackets indicate the construction of the two diatonically disjoints tetrachords; they are the fundamentals to build the heptatonic scales of the major-minor tonal system. By transposing the three previous modes over the twelve chromatic tempered notes, we built the whole set of scales of our considered tonal system. We end up here with 36 scales.

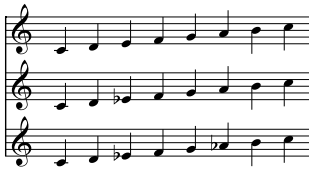


Figure 1. The major, minor melodic and harmonic minor modes of the tonal system (here given in C)

1.2 Definition

The notion of tonal signature starts with a simple idea. In the context of western classical and improvised music, the tonal system plays an important role. To a certain extent, from free jazz to main stream improvised music, and towards experimental improvised music fields, it is often possible to use extended scales that are still making reference to the notion of center or tonality. Depending on the playing, on the speed and on the compositions, different degrees of complexity give a more or less clear readability, or intelligibility, of the tonal center, even if modified by fast modulation changes. Malik Mezzadri has been looking for a minimal principle to deal with the readability of this center, something that would act as a minimum of information needed to understand the tonal center for someone who does share this cultural and musical background. Shouldn't a principle of minimalism also imply simplicity? The Tonal Signatures embrace the idea that the French flute player was initialy expressing and experimenting intuitively with his musicians in his compositions, before formalizing it. We define the tonal signature concept as the following :

Definition 1 : *A tonal signature is the smallest set of notes which belongs to one and only one transposition of a mode of the major-minor tonal system.*

Definition (1) makes this group of notes, by the condition of unique existence, and if proved, an exclusive *signature* of the considered transposed mode, a given scale, among the whole thirty six possible scales. A tonal signature does not contain enharmonic ambiguities, which means that the set of sounds handled in one tonal signature is also unique and specific of the mode and its own specific transposition. We can reformulate definition (1) in a slightly more mathematical way :

Definition 2 : *A tonal signature of a scale S is a minimal subset of notes within S that is not contained in any scale S' different from S.*

Let us explain why the notion of minimality makes sense. For a given scale S, a *typical* set T of S is any subset of S that is not contained in any other scale different from S. Obviously, any set T' such that $T \subset T' \subseteq S$ is also *typical*. So, it makes sense to look for *minimal typical* sets. Such sets are what we call tonal signatures. The main questions we now have are : How many tonal signatures do exist ? How do we find them ? How can we use them for composition and improvisation ? Can we extend the tonal signature concept with other musical modes ?

1.3 Finding tonal signatures

We use the computer assisted compositional environment *Open Music*[1] to look for the solutions of the tonal signature problem. To elaborate the research of the minimal *typical* subsets, we define some constraints (Figure 2) to generate the subsets of notes among the 36 tonal scales defined in §1.1 and to look for unique belonging property to the reference scale (Figure 3). For example, if we consider the scale of C major as the reference scale, we want to find what are the smaller and typical subsets of notes that only belong to C major. We used the *Screamer* [1] constraint solver to express the constraints and find the solutions.

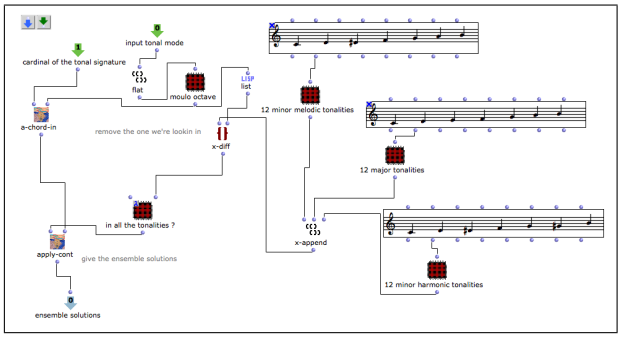


Figure 2. constraint definition in *Open Music* with *Screamer* solver to find a tonal signature

We end up with seven tonal signatures (Table 2) with their structures in semi-tones related to the given tonic. In Table 2 and note examples are given for the C tonic.

We use the following convention for names : *M* for the major mode, *h* for the harmonic minor mode and *m* for the ascending melodic minor mode, followed by a figure before the tonic letter to label the solution if more than one.

The *Screamer* constraint solver builds its own search tree and we don't have too much control of how it looks

[1] Screamer is an extension of Common Lisp that adds support for nondeterministic programming. Screamer consists of two levels. The basic nondeterministic level adds support for backtracking and undoable side effects. On top of this nondeterministic substrate, Screamer provides a comprehensive constraint programming language in which one can formulate and solve mixed systems of numeric and symbolic constraints.

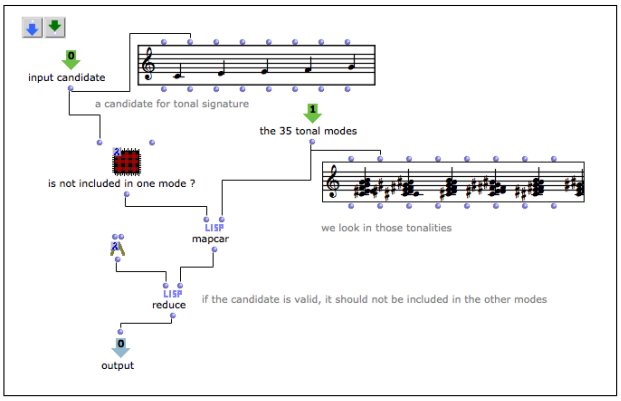


Figure 3. Constraint rule expressing unique belonging property of a candidate signature in the set of scales

Table 2. the 7 tonal signature structures

Modes	*Structure*	*C tonal signature*	*name*
Major	$0, 4, 5, 7, 11$	C, E, F, G, B	MC
minor	$0, 3, 5, 9, 11$	C, Eb, F, A, B	m1C
	$3, 5, 7, 9, 11$	Eb, F, G, A, B	m2C
	$2, 3, 9, 11$	D, Eb, A, B	m3C
harmonic min	$0, 3, 8, 11$	C, Eb, Ab, B	h1C
	$0, 7, 8, 11$	C, G, Ab, B	h2C
	$2, 7, 8, 11$	D, G, Ab, B	h3C

for the solution. But as it gives us the solutions and because the set of scales and subsets is not too large, we can cope with this. We are implementing a new version with the *Gecode* [2] constraint solver as this library is now available in *Open Music* and also in the *C language*. This is interesting as we can also use the solver motor in a real time environment like *Max5* [3] to request solutions on demand with nearly no time headroom. This option has also the advantage to enable the use of other scales and constraints as we will see later.

It is nonetheless possible to prove mathematically the existence of the tonal signatures by considering the problem as an appropriate set covering model [2] which belongs to the class of integer linear programming problems. Arbitrary set covering problems are NP-complete [3], but [4] prove in this more general case that a fast and direct algorithm solves the tonal signature problems.

2. A COMPOSITIONAL APPROACH

We now focus on the use of the tonal signatures. Tonal signatures provide both material for composition and for improvisation. To stay with the same minimalistic approach as their definition, we hardly ever use the tonal signatures $m1$ and $m2$. $m3$ is a smaller signature so we privilege this one and skip the larger ones. In this compositional context we only use five tonal signatures : $M, h1, h2, h3$ and $m3$, and we now notate the minor tonal signature $m3$ as m (Table 3).

[2] `http://www.gecode.org/`
[3] `www.cycling74.com`

Table 3. context of 5 minimal tonal signatures

Modes	*Structure*	*C tonic*	*name*
Major	$0, 4, 5, 7, 11$	C, E, F, G, B	MC
minor	$2, 3, 9, 11$	D, Eb, A, B	mC
harmonic minor	$0, 3, 8, 11$	C, Eb, Ab, B	h1C
	$0, 7, 8, 11$	C, G, Ab, B	h2C
	$2, 7, 8, 11$	D, G, Ab, B	h3C

2.1 Some musical implications

One of the first implication of the use of tonal signature is the diminishment of the functional possibilities of the harmonic material of the tonal system. The tonal system is strongly rooted in the western culture, and we can understand easily it without going through the abstract or philosophical ideas of its evolution. A function from the tonal system is a musical construction that is inclined to underline or undertake the reference system in which the music takes place.

The favorite subject of tonal system is itself. The music that grew up in this context organized itself in structures and tools as a tautology of its own views. The tonal system is indeed almighty, and organizes the musical materials so as to maintain its supremacy. It is a fact in the classical period that even the more subtile harmonic transitions are speaking by themselves of the tonality of the composition. And so it is inside the western cultural musical background, people became soaked with tonality and knew it intuitively, unconsciously : for a non musician, it is often not difficult to organize a small melodic phrase that makes reference to a tonal mode. It is not new that the tonal system creates a hierarchical music where no subset of the scale can pretend to the same rights as the whole reference scale has.

With the tonal signatures, we nevertheless achieve the building of subsets that do share the same functional prerogatives as their ensemble of reference : being clearly identifiable to their reference scale. The C major mode for example, is nothing else than the C major mode within the tonal system, whereas a singled out triad of C major would not be evidence of the C major mode, as it could be evidence of another mode. We need a combination of several triads to recognize what is the C major mode. The MC tonal signatures is enough to restore this information. In fact, any tonal signature restores the identity of the mode it belongs to. As it is now a minimal information, we might nevertheless find ourself a bit lost or disturbed to the listening to some incomplete parts of a scale, even if they should retain the whole information of the mode ! We might explain that by the fact that the tonal signatures retains, as described in §1.2, the *minimal* and *typical* melodic structure of a tonality, but implicitly is retains less of its harmonic structure. We still have the minimal information to understand the notes belonging to a mode, but we have less information to restore all the harmonic qualities of the same mode among the whole tonal system.

2.2 *Resolutions* with tonal signatures

With the tonal signatures, we create a paradoxical situation where we hold harmonic ambiguities, or rather non complete harmony, together with unambiguous melodic information. In this new situation, we are strangely in between modality, from melodic consideration, and tonality, from harmonic consideration. In this context, we propose two ways to use the tonal signatures to compose musical forms : one is an afferent or inward *resolution*, the other is an efferent or outward *resolution*.

2.3 Afferent movements of the tonal signatures

We mean with afferent movements to use the tonal signatures in their own particularity but retaining the link they have with the tonal system. We can make a comparison with the dodecaphonic serialism attitude : Arnold Shoenberg often used his musical material in afferent way, inserting it in forms and structures of the tonal system. Conversely, Anton Webern deduced forms and structures from the musical materials and its inner rules, as if the form was a consequence of the dodecaphonic series and its inner arrangement. We note that the *functional* use of the tonal signature inside the tonal system claims a kind of serial existence : to achieve its *function*, a tonal signature must be explicit and all of its note have to be used.

2.3.1 Common notes : sequence vs progression

The project and record [5] has been realized with the afferent techniques use of the tonal signatures. The approach was to use tonal melodies from the cultural repertoire and to harmonize it only with tonal signatures. To achieve this, we match the mode the parts of the melody makes reference to, and we proceed to the writing of the sequence of tonal signatures. We insist on the term *sequence* and not *progression* : *progression* is like chord changes and would emphasize or point a tonality within the *chain* of the musical material, while the term of *sequence* implies to keep the maximum of common notes in respect with the given melody. Besides, to keep a musical touch and feeling of the harmonization, choices are done intuitively so that the foreign notes brought implicitly or *per se* with the tonal signatures don't disturb too much the melodic constraint. The first musical tunes written with this system are *"Z"* in [6] and *"Vienne"* in [6]. The former undertakes harmonic realization of the tonal signatures, while the later undertakes melodic realization; the original harmony is of a classical tonal implementation.

2.3.2 Tritone resolution

Another possibility for afferent use of tonal signatures is triton resolution. This time we can consider progressions, as we deal with functional quality of the chaining tonal signatures. We build progressions on the tritons resolution inside a tonal signature towards the next one, so that they also share, or not, the most possible common notes with the tonal reference of this resolution. We have considered the two easier resolutions with semi-tones. For example $(E\#, B) \Rightarrow (F\#, A\#)$, and $(F, B) \Rightarrow (E, C)$. A composition where this principle is used is [hidden for blind

reviewing] in the project [hidden for blind reviewing] [7]. Here is, from the (F, B) triton and its tonal en-harmonies, the set of tonal signatures that contain its resolutions :

1. $MF\#$, MC
2. $m1C\#$, $m2C\#$, $m3G$, $m3Db$
3. $h1Bb$, $h1E$, $h2Bb$, $h2E$, $h2B$, $h2F$, $h3B$, $h3F$

We got interested in applying tonal signatures to the triton resolution because it is a subject of discussion since ages, and as such is a subject of importance that cannot be ignored.

2.4 Efferent mode of the tonal signatures

We develop here other ideas to use the tonal signatures. The tonal signatures are coming from a tonal context, from a deductive and *conservative* approach, which involves *minimal* means to retain functional data relative to the three subsets of the tonal system. But the tonal signatures are not so familiar to listen to. If we play an $h2$ tonal signature, even looping this *information*, it would still be difficult for an average listener to identify it with certainty to its reference mode. We can nontheless make the most of this remark! We can use this material in a efferent way, knowing that the *information* it holds would have few or maybe no interferences with other *information* from another context. Moreover, the tonal information of a tonal signature would also be restored in a new context, in a more subtle way which can stimulate a lot the cultural memory of the listener.

2.4.1 Talea color

The choosen approach in this project is realized in reference to the *Talea Color* of the *Ars Nova* period. The melodic material or *color* is rhythmically organized with a given temporal cycle or *talea*.

In the first record of [6], the tonal signatures are used as a polyphonic material without contrapuntal treatment, but rather like monoliths. The score is built from a bass line which acts as an attractor for the tonal signature, chosen for its harmonic role or in respect to common notes. We can build this way a polyphony of tonal signatures, which makes a musical topology on top of which improvisation can unfold. Rules for music ranges are added to constrain the scoring, and these combinations tends to make the composition, as an example is given in (Figure 4), resemble a kind of cellular automata process .

2.5 Tonal signatures among Messiaen's modes

The meeting with Olivier Messiaen's modes of limited transposition is very stimulating. They are a main domain of application with tonal signatures in the efferent mode. The limited transposition modes of Messiaen, notated *LTMM*, are a logical consequence of the tempered system, maybe even more than the dodecaphonic system. We can see Messiaen's modes as a gradual chromatic development of the tonal system. The *LTMM* are subsets of the chromatic scale, and because of their inner symmetry, they remove

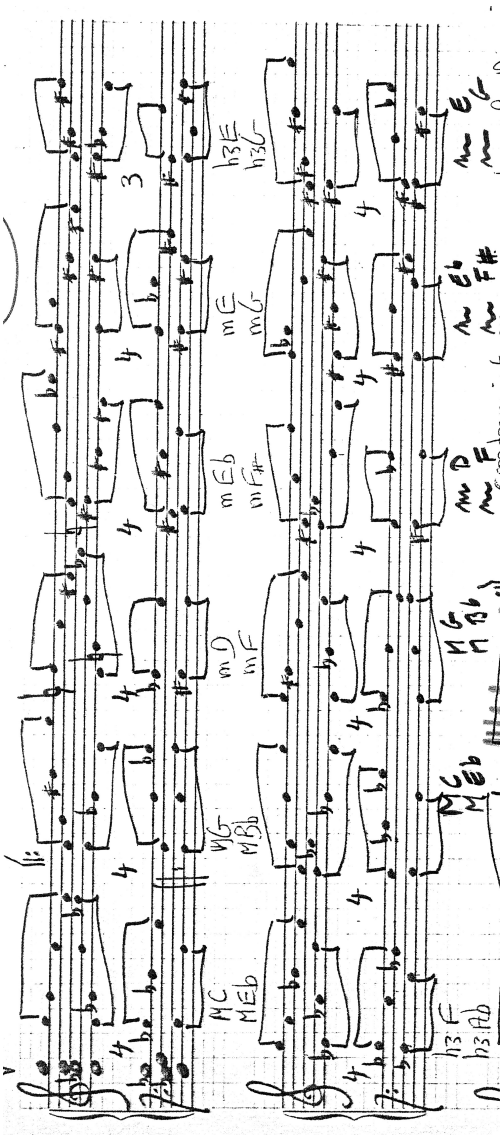


Figure 4. polyphony of tonal signatures within constraint of tessitura given as a chord at the start of the stave

polarity of notes, as found in tonal system, and replace it with polarity of axes. From the seven *LTMM*, we only consider the third and the seventh mode (Figure 5), as they contain all the others, considering all the transpositions. If we look for tonal signatures in limited transposition modes *M3* and *M7*, we now find the same intervalic structured signatures in one mode, i.e. at least one transposition of the same signature is found in the modes (Table 4). From this very singular observation, we can now consider the possibility to go back to the idea of progressions, and not merely sequences. We can reformulate the same idea : from the tonal system context where we have tonal signatures with notes, we arrive in the *LTMM* context to sign progressions with tonal signatures, that we can call *progression signatures*.

3. CONCLUSION AND FUTURE PERPSECTIVES

Mixing tonal signatures concept within context of Messiaen's modes is opening some new material and intuitions

for new compositions and improvisations. We can use the tonal signatures also in more conventional forms, as *Canons* or *Counterpoints* (Figure 6). We will remind that in Messiaen's mode context, where there is by symmetric construction no more tonic function, the tonal signature concept fits very well, because it does not make a classical hear reference to one scale, as sometimes it also misses tonic or triad. Then the real difficulty begins for the musicians, as one needs to build reflex and a new musical thinking mind with this musical material, to digest it, so as, from a new compositional language it becomes also a natural language to improvise and play with.

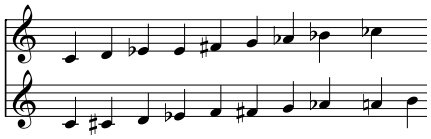


Figure 5. Limited transposition modes of Messiaen : CM3 and CM7

Table 4. Tonal signatures in Messiaen's mode

LTMM	*Tonal signature inside LTMM*
CM3	$MEb\ MG\ MB\ m1Eb\ m1G\ m1B$ $m2Db\ m2EB\ m2F\ m2G\ m2A\ m2B$ $m3Db\ m3F\ m3A\ h3C\ h3E\ h3Ab$ $h1C\ h1Eb\ h1E\ h1G\ h1AB\ h1B$ $h2C\ h2Eb\ h2E\ h2G\ h2AB\ h2B$
CM7	$MDb\ MD\ MG\ MAb$ $m1C\ m1D\ m1EB\ m1F\#\ m1Ab\ m1A$ $m2C\ m2D\ m2E\ m2F\#\ m2Ab\ m2Bb$ $m3C\ m3Eb\ m3E\ m3F\#\ m3A\ m3Bb$ $h1C\ h1Eb\ h1F1\ h1A$ $h2C\ h2C\#\ h2F\#\ h2G$ $h3C\ h3C\#\ h3E\ h3\#\ h3G\ h3Bb$

4. REFERENCES

[1] G. A. Jean Bresson, Carlos Agon, "Openmusic : A cross-platform release of the computer-assisted composition environment." Ircam, 2003.

[2] B. Simeone, G. Nouno, M. Mezzadri, and I. Lari, "A boolean theory of signatures for tonal scales," rapporto tecnico 13/2010, Dipartimento di Statistica, Probabilita e Statistiche Applicate, Sapienza Universita di Roma, 2010.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NPcompleteness*. New York: W. H. Freeman and co., 1979.

[4] B. Simeone and I. Lari, "Boolean harmonies and a tractable class of set covering problems," raporto technico 14/2010, Dipartimento di Statistica, Probabilita e Statistiche Applicate, Sapienza Universita di Roma, 2010.

[5] M. M. Orchestra, "13 xp song's book." Label Bleu, 2006.

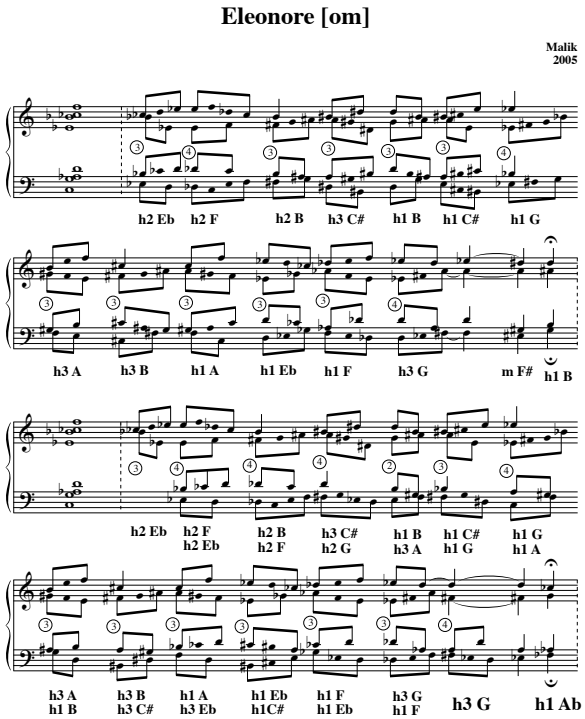


Figure 6. Four voices counterpoint, polyphony of tonal signatures mixed within Messiaen modes's context

[6] M. M. Orchestra, "00237-xp1." Label Bleu, 2007.

[7] Octurn, "Xp 26." Octurn XP live, Octurn production, 2008.

ACOUSTIC REHABILITATION OF A LITTLE CHURCH IN VINAROS (SPAIN)

Jaume Segura
IRTIC – Computer Science Dpt
Universitat de València
jaume.segura@uv.es

Álvaro Romero
Dpt of Musicology
Universitat Catòlica de València
alvaro.romero@ucv.es

Enrique Navarro
IRTIC – Applied Physics Dpt
Universitat de València
enrique.navarro@uv.es

ABSTRACT

Some old churches in Spain are presently used for concerts and other musical or leisure performances. The acoustical conditions of theses buildings are not optimal for these new uses. For this reason, it is necessary to modify its original structure in order to achieve better acoustical features. This paper describes the work done in order to improve the acoustics of a church in Vinaròs, it is presented the refurbishment of this space as a multiple-use room, also to keep the old aesthetics it is proposed a reversible actuation. The proposal is virtually analyzed by using a ray tracing tool.

1. INTRODUCTION

Acoustic rehabilitation of churches for a non-liturgical use is an open field in acoustics. Many resources and efforts have been invested to obtain the adequate public environment where anyone can listen to a specific kind of musical signal. In order to develop this sort of rehabilitation projects several techniques can be used. Among them the most well-known techniques are: the method of the scale model for the acoustical modelling and acoustic simulation techniques by means of several numerical methods. Here, it is interesting to point out that the acoustic simulation is quite cheaper than other methods, for this reason nowadays they are widely spread.

Old Churches are presently used as concert rooms, however the structure of these buildings is not optimal. The modification of these churches for non-liturgical uses have made grown interest for rehabilitation of these ecclesiastical spaces, in order to adapt them as polyfunctional rooms devoted to cultural or leisure activities.

In the Iberian Peninsula, most of the towns and cities have one or more churches, chapels, cathedrals, monasteries or convents, which have had modifications inside through the centuries and, therefore, have varied their acoustic conditions. Thus, when they are refitted for other uses, as for example musical auditoria, it would be interesting to take into account this factors, in order to have similar final results as in the original case, since the acoustical conditions of the churches are deficient for new uses in most of the cases, due to architectural alterations, changes in the furniture, or decoration, it has also influence the different position of musicians, performers and audience.

In this rehabilitation proposal, the original structure of the church as a whole has been respected. Also, the aesthetic and artistic identity has been preserved, acting in a reversal way and making adaptations for each one of the needs and uses of the room. From the point of view of conservation and refurbishment of patrimonial goods, we have respected the original space, but with minimum changes. From the point of view of the current use, we have proposed a set of reversible alterations in the environment which allow optimum acoustical conditions in this room for the uses planned. In this paper, a simulation with CATT Acoustics simulation package has been made, in order to predict the response of the acoustic refurbishment proposals.

2. HISTORY AND CHARACTERISTICS

Saint Augustine Church is part of the remains of the old Augustinian Fathers Monastery which dates from the 16th century. It is a baroque building that has a Latin cross shape with three side chapels at each band among the buttresses connected themselves. The stone cross has a dome upon the tambour and scallops and a presbytery roofed with groin vault, as well the side chapels of the nave, which are nowadays totally empty. The façade stands out by its symmetric composition. At both sides of the entrance two bell towers are risen with their own door, which frames the central wall crowned by a molding cross section and its façade of lintel access (Fig 1).

Figure 1. Saint Augustine Church façade.

The church was used as a stable for the Napoleon cavalry in 1808 and also it was deteriorated during the Spanish Civil War. The church was used temporarily to worship and it was definitively closed to the audience in 1975. In the 80's (eighties), the old church was fitted out and re-

stored functionally as an auditorium, being nowadays the Wenceslao Aiguals de Izco Local Auditorium
Saint Augustine church consists of a 400m^2 surface, which has attached the chapel of Saint Victoria in which the Local Museum rooms are located. The plan church is 29.3 meter long from the entrance door to the apse; it is around 16 meter wide counting the width of the side chapels, 13.2 meter high from the floor to the vault and 23.1 meter high to the central part and the highest from the dome. Therefore the church volume is around 4900m^3. The stalls have a surface of 72m^2 and consist of 14 rows of 13 seats each one of them (Fig 2).

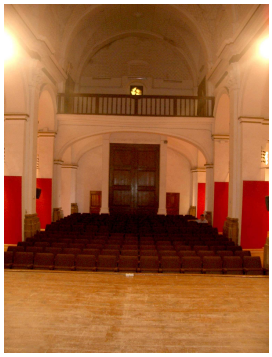


Figure 2. Stalls and choir of the church.

3. MODEL AND SIMULATION: A PREVIOUS STAGE

3.1 Geometric model

In order to model Saint Augustine Church in Vinaròs, a variation of the ray-tracing method has been used. This was a hybrid cone-tracing combined with an image-source approach called randomized tail-corrected cone-tracing algorithm (RTC-II). The geometric model was developed by using CATT Acoustics [3], in order to predict the acoustic behavior and to carry out a valid global proposal to its restoration and acoustic fitting-out.

Ray tracing algorithms take into account that a wave generated by a source can be linked to a ray, cone or pyramid by means of the eikonal equation. The source statistically emits a series of rays with a specific energy which, on inciding with the surfaces of the walls which delimit the geometric model, lose energy with each reflection. In our case, both calculation programs used take into account specular and diffuse reflections. In the case of CATT, the diffuse reflections are calculated from the explicit parametrization of the absorption and scattering parameters of each surface.
The modeling process of one court consisted in defining all surfaces in the same court, whatever its shape and dimensions, specifying the coordinates of its respective vertexes [4]. In the case of Saint Agustine Church of Vinaròs (as has already explained above), the coordinates have been calculated from the ground plan provided by the Culture Department of the Town Council of Vinaròs and from the measurements made with an ultrasound sensor of the elevation levels necessary to develop the model in three dimensions that allows simulating of the acoustic behavior of the court and, in this way, can put in to carry out the proposal of adequate materials to the restoration and acoustic fitting-out of the church to can be used as an auditorium (Fig 3).

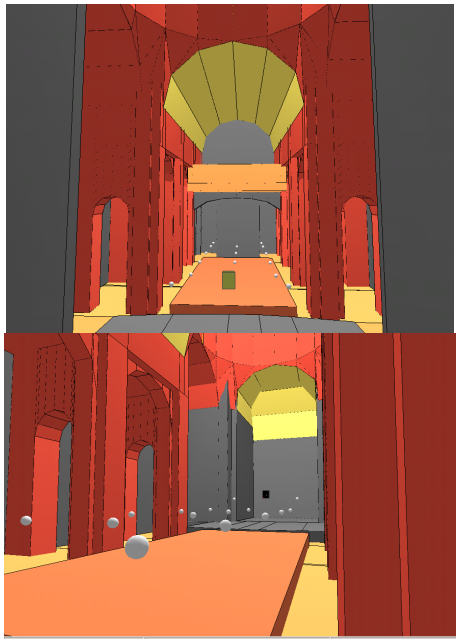


Figure 3. Different views of the modeled church in CATT. *(a) front view, (b)back view*

As each one of the surfaces the church consist of a certain material, it is necessary to provide the respective values of absorption coefficients of each one of them. As surfaces are smooth enough, their scattering coefficients are low, (about 10%) according to the Lambert law. Finally, to complete the model, it is necessary to specify the position and characteristics of the sound sources in the building and the positions of the respective receivers.

3.2 Calibration of the model

In order to make a proper calibration of both models, measurements were made by using a MLSSA card, which uses the MLS technique to perform the acoustic assessment of parameters in any room according to ISO 3382-1:2007. Measurements were done in 16 positions.
The MLS technique uses a pseudo-random binary sequence as excitation, whose self-correlation function corresponds to a Dirac delta ($R_{xx}(t) \approx \delta(t)$). The cross-correlation of any signal with $\delta(t)$ is the signal itself, which means that:

$$R_{xy}(t) \approx \delta(t) * h(t) = \int h(\tau)\delta(t+\tau)\, d\tau = h(t) \quad (1)$$

Therefore, in a linear system, if we calculate the cross-correlation between the MLS signal applied to the input and the signal recorded at the output, we can determine the impulse response of this system and thus calculate its transfer function by means of applying the FFT.
The values obtained for RT30 in octave frequency bands were used to calibrate both acoustical simulation models. In this case, an error up to 10% in this reverberation parameter has been allowed, in order to keep the 'just noticeable differences' (jnd's) values in an adequate range for all the parameters. Figure 4 shows the spatial average values per frequency bands of RT30 for the tonal curve of

the room. The error bars show the difference between measurement and simulation.

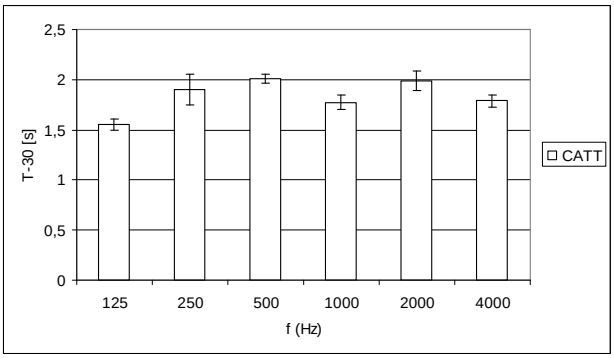


Figure 4. *Average tonal curve for T30 measurements, CATT for 16 positions*

Once the models are developed and calibrated, they are available for throwing a certain number of rays (with a number high enough to get valid results according to the volume of the room). These rays come from the sources modeled in the process.

In our case, 20,000 rays were thrown, considering for CATT 2 seconds of truncation time. This choice presented a compromise formula obtaining the satisfactory results for a reasonable calculus time. After tracing the whole amount of rays and processing the information of the simulation, it is possible to obtain the impulse response in a specific receiver [5], [6].

The original church is modeled, before intervening (sticking absorbent panels in the vault, apse, etc). The surfaces are defined and are organized in groups, attributing the type of material they are made up of to each surface or group. Five materials have been considered mainly in the original church. The configuration of materials to model the original church has parquet floor for the stalls floor, which show a slight slope in order to get higher height. In the side chapels, the central round, the sides of the church and the front and back walls of the room, the material used is plaster and vermiculite. In the vaults and dome, the chorus, the entrance door and the stage floor, common wood was used. For side corridors and for the ones which are between the stage and the first row of stalls carpet was used. The values of the respective absorption coefficients (α) have been taken from the bibliography [7] for frequencies between 125 and 4000Hz in octave bands.

3.3 Sound source

The characteristics of the source have been defined as well as its position and orientation that could belong to a person's mouth located in the centre of the stage, in the position (X;Y;Z) (19.2, 7.9, 2.3) with a certain power. To these effects and subsequent ones, it could be noticed that the origin of coordinates (0, 0, 0) chosen to model the court have been located at floor level at the left side of the entrance of the church seen from the stage. The acoustic characteristics of the source are:

Position: X = 19.23 Y = 7.90 Z = 2.30						
Orientation: Azimuth = 0.00º						
Rise = 0.00º						
Rotation = 0.00º						
Limit angles: Azimuth: between 0.00º and 360.00º						
Rise: between -90.00º and 90.00º						
Delay in the emission: 150 ms						
Acoustic parameters (to the six octave bands between 125 and 4000 Hz)						
Frequency	125	250	500	1000	2000	4000 Hz
Resonant level to 1m	70.0	73.0	76.0	79.0	82.0	85.0 dB
Power	73.1	76.1	79.1	82.1	85.1	88.1 dB
Horizontal -3dB : 35.00º (to all octave bands)						
Vertical -3 dB: 45.00º (to all octave bands)						

Table 1. Sound source characteristics.

Regarding to the number of receivers, 16 were considered regularly distributed in the stalls, three receivers in the first seat stall (a), in the seventh and thirteenth of the first row, in the fifth, ninth and fourteenth. Two detectors were positioned symmetrically on the stage at both sides of the source (Ed and Ei, right and left stage) and two detectors more in the choir, to both sides (Cd and Ci, right and left choir). Figure 5 shows the positions of each one of the 16 receivers considered, as well as the source position.

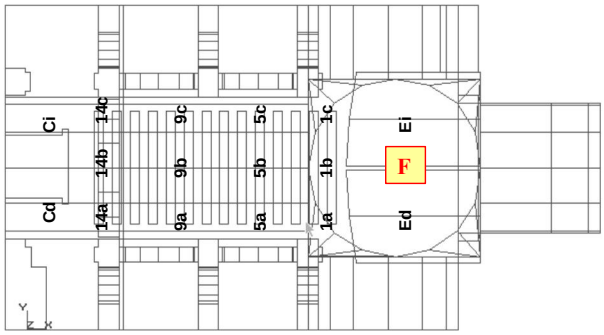


Figure 5. *Average tonal curve for T30 measurements, ATT for 16 positions*

The number of the receivers used is big enough to characterize the acoustic response of the church in detail and has been selected according to the ISO 3382-1:2007. It is also necessary to introduce data to the software (CATT), because of the propagation properties in the environment such as: relative humidity (50 %), room temperature (20 °C) and air absorption (dB/100m). Finally, it is interesting to emphasize that to build the model of the Saint Agustine Chuch, 530 different surfaces were considered.

3.4 First calculations

After the analysis of the results obtained from the predictive running of the model of Saint Agustine Church in its original state, it should be pointed out that the church offers poor acoustic conditions for talking and for musical and multi-functional use.

On one hand, the reverberation time ranges obtained are between 1.35 and 1.95 for CATT. The optimum reverberation time for a court whose volume is 4900m^3 should range between 0.9 and 1.3s [4], [8], [9], [10], [11]. This reverberation time is produced by a high reflectivity and low diffusion on the walls and the plastered ceiling (and *vermiculite*). We also have to keep in mind possible fo-

calizations, which can appear due to the dome, the vaults and the apse. All these focalizations result in a non-constant distribution in the room, as in case of the pressure levels and the definition and clarity indexes (D50 and C80).

In this way, the pressure levels off in the room after a typical distribution of decreasing in the receivers each time further away from the source. This is due to the decrease of intensity with the distance, in free field conditions. For this reason, reflections should be reinforced in the last rows, creating a diffuse field, more uniform in the whole audience zone and the stage. It is observed that the pressure levels decrease up to 10 dB between the receivers located in the stage and located in the last rows.

The values for the definition D50 index are included between 30% and 72% for CATT. These values adequate, but ideal values should be taken between 50% and 75% to assure good intelligibility. In this way, speech does not loose definition or clarity and it can be listened in a proper way in every place of the church [4]. Therefore, the values obtained at the stage receivers oscillate between 65% and 78% for CATT. These high values for D50 parameter are reasonable due to the proximity to the source. In the same way, analyzing the values of the clarity index C80, we can conclude that the results are not the optimum ones. Although the values for the front rows can be considered as acceptable, the results for the back rows are totally unacceptable [4].

For Speech Transmission Index values (STI) and Rapid Speech Transmission Index (RASTI), the results obtained qualify the church acoustically in its original state as a weak or regular room. The values of the STI index were between 0.5 and 0.66 for CATT. These indexes are measured in normalized units between 0 and 1 and show the intelligibility in the church [8], [4], [9]. The results obtained show regular intelligibility to use the church as a room for speech.

4. RECOMMENDATIONS FOR REHABILITATION

The main aim exposed in the study of Saint Agustine Church in Vinaròs is to improve the acoustics of the room, although respecting the structure and aesthetics of the church as much as possible. Here it is important to point out that the church in its current state has already undergone a process of fitting-out. This is the main reason for the existing slight slope in the stalls, as well as the use of parquet flooring, the carpet and the absorbent material in the vault, the dome and the apse. The absorbent material was damaged in its function, as well as aesthetically, due to humidity. In this work, this absorbent material has not been considered in the modeling, but instead the church has been taken in its original state, without this absorbent material. The slope of the stalls, the parquet and the carpet of the floor have also been taken into account. Non-considering the absorbent material is due to aesthetic reasons, as the absorbent material panels break the aesthetics of the building. Therefore, in the global proposal of rehabilitation and acoustics fitting the preservation of the original structure has been contemplated first working on it in a reversible way. This was made by implementing several objects which can be removed after the performances, if desired, giving a wide range of possibilities.

After the study of the several proposals made in several churches [8], [12], [13], [14], [15], [16], [5], [6], [17] and the analysis of the different possibilities for Saint Agustine Church, our proposal consist of the following corrections in its model:

- The sloping floor of the stalls is kept, but the floor material is changed into an ordinary/poor-quality wood floor; the lateral corridors and the front corridor of the stalls are made of carpet; the floor of the chapels and the stage corridors are parquet; the walls of the chapels, apse, domes, vaults, walls and friezes are still made of plaster and vermiculite as in the original modeling; the balustrade of the choir and the access door to the church are made of wood.
- The setting of mobile curtains of gathered velvet is modeled, in front of the arches of the side chapels and covering the base of the dome, in such a way that it is not only used as an absorbent material but also it provokes a higher number of reflections on the audience zone, avoiding the fitting connection to the chapels. The focalization coming from the rays reflected in the dome and the delays or focalizations not desired, which increase the reverberation time punctually.
- At the end of the stage four common wood panels are set out, making up an acoustic shell that is located in front of the apse. These panels can be provided with wheels in order to carry them. The panels (Figure 6) are around 7 meter high and around 1.74 meter wide, which remain in the centre (a) and 1.86 meter wide, which are outward oriented (b). With this measurement, the reflections coming from the apse and its possible focalizations are mitigated. It also allows for the early reflections for the audience, as well as for the performers who can be listened to themselves better.
- On the stage three sheets/plates of stained-glass are hang up with several slopes, making up another acoustic shell (Figure 6), so they do not hamper the vision, but the sound arrives before to the audience as the reflections are produced in full time and are directed to the audience in a more direct way.
- Moreover, in the modeling a fourth stained-glass sheet/plate is setup at the feet of the choir, under the balustrade at the access door. This plane has a slope which produces reflections of the sound rays on the stalls, fostering the first reflections in the last rows. The sheets have different dimensions: the three settled on the stage are about 7 meter wide and 1.91 meter (c), 3.10 meter (d) and 2.00 meter (e) deep; and the choir sheet/plate has the dimensions of 1.20 x 6.40 meter (f).

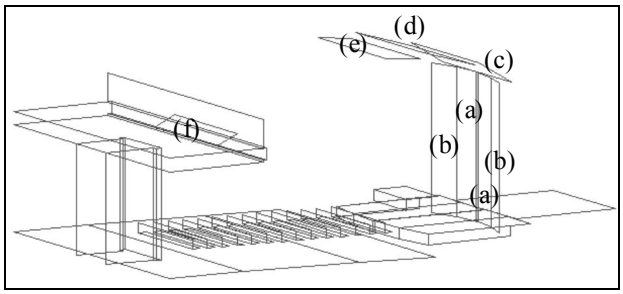


Figure 6. *Diagram with wood and stained-glass sheets.*

A change in the seats at the stalls is modeled that in the previous model for the acoustic fitting-out proposal was covered, including padding of half type, which is replaced with padding stall seats of bigger sound absorption. With this measurement, the acoustic absorption of the stalls is increased, where the reflections have been directed. This is the place where the audience is located. The number and rows of the stall seats remain as in the original state: 14 rows of 13 stall seats in each one of them.

In Table 1, the absorption values (α) for the octave of frequency between 125 and 4000Hz of the materials considered in the acoustical rehabilitation have been taken from the bibliography [7]. Also the surface used for each considered material is shown.

Material	Surface (m^2)	Absorption Coefficient α					
		125	250	500	1000	2000	4000
Plaster	*2898.04*	0.120	0.100	0.070	0.090	0.070	0.040
Ordinary wood	*592.65*	0.200	0.160	0.130	0.100	0.060	0.050
Parquet floor	*280.59*	0.200	0.150	0.120	0.100	0.100	0.070
Stained-glass sheets	*60.96*	0.180	0.060	0.040	0.030	0.020	0.020
Upholster stall seats	*73.43*	0.360	0.430	0.470	0.440	0.490	0.490
Carpet	*356.80*	0.130	0.060	0.130	0.200	0.460	0.700
Gathering velvet	*233.69*	0.070	0.310	0.490	0.810	0.660	0.540

Table 2. Absorption coefficient in frequency bands for each material.

The source and the receivers are previously considered, with their same names and characteristics. Once the church is modeled with the proposal of rehabilitation and the acoustic fitting-out proposal, we proceed to carry out the simulation of the room and to obtain the calculation of the acoustic parameters which characterizes the room. The simulation is made with a total amount of 20000 rays, taking into account in each case and setting a truncation time of 2 s in CATT.

5. RESULTS

The results obtained are shown below in a comparative way for the model simulated with CATT Acoustics [3] of Saint Agustine Church in Vinaròs in its original state and after applying the general proposal of restoration and acoustic fitting-out. Firstly, the calculated reverberation time is shown in three different ways (Sabine, Eyring and general or statistic), the level of the resonant pressure, the indexes of the D50 definition and C80 clarity, the STI and the RASTI parameters.

5.1 Reverberation time

The global reverberation time of the church is obtained in three different ways, taking into account the Sabine formula, the Eyring formula and the statistic way.

The statistic reverberation time before the restoration and acoustic fitting-out proposal is between the values 1.49 and 1.88s. Therefore, the reverberation time of the original church is high for the uses that we want. Moreover, the reverberation time at low frequencies is lower than the medium ones. In this case, a higher RT at low frequencies is needed in order to improve the Bass Ratio (BR) in the room. Also the statistic reverberation time proposed takes values between 0.96 and 1.36s. Therefore, the reverberation time of the church once carried out the restoration and acoustic fitting-out proposal would be the adequate, as these values would be very near the intervals from 0.9 to 1.3s. The reverberation time in low frequencies is also higher than the medium one, around 20% therefore this good fact contributes to give more acoustic warmth to the room. In the following figures 7 reverberation times are shown:

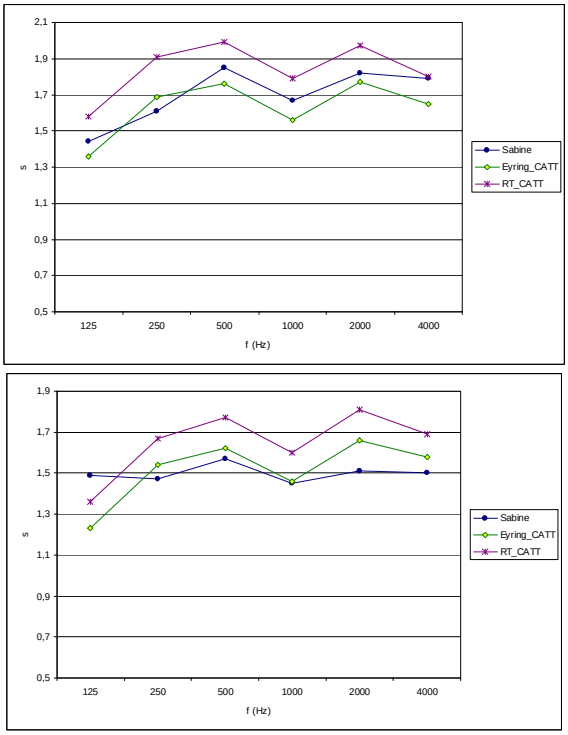


Figure 7. *Reverberation Times for "Sant Agustí" before (a) and after (b) the proposal.*

In the three cases, the reverberation time improves after the application of the proposal for rehabilitation and acoustic fitting-out. Reducing and adapting better to the optimum reverberation time of the church implies that it would be between 0.9 and 1.3s. So, in this way, the reverberation times before the proposal are between the

values 1.31 and 1.88s and after it between 0.80 and 1.36s which confirms the improvement.

5.2 Pressure level (dB)

The pressure level L_p (dB) also experiments an important improvement after the acoustic fitting-out proposal, so it changes to adopt a more uniform distribution. In the former case, the pressure field has a similar behavior as in the free field. After the proposal, the field changes to be more homogenous in all the room. Therefore, pressure changes to adopt a diffuse field condition. This is shown in the values obtained for the 16 receivers, which ranges between 71.7 dB and 61.5 dB before the proposal and between 77.0 dB and 69.0 dB after carrying out the proposal, but also in the Sound Pressure Level distribution maps in Figure 9.

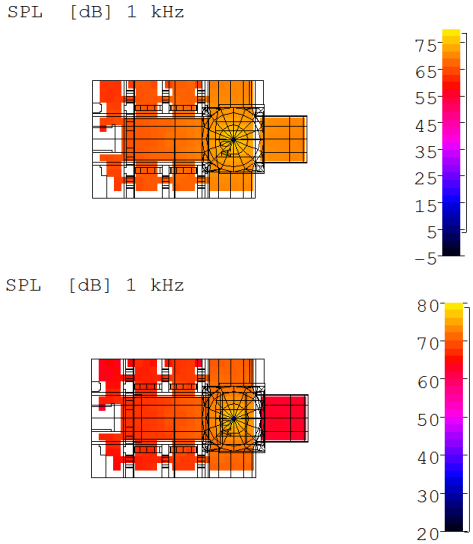


Figure 8. *General Pressure Level Distribution before (up) and after (down) the proposal*

5.3 Definition D50 (%)

Regarding the definition index, the changes in the results obtained before and after the restoration and acoustic fitting-out proposal are also noticeable. They do not take values between 12% and 43%, but instead between 39% and 79 %, which provide the church with better speech intelligibility. A room with these features can take, in the ideal case, values between 50% and 75%, what is achieved in most of the receivers, except from those in the middle part of the stalls. Figure 10 shows the maps of definition index D50 distribution for the 1000Hz band.

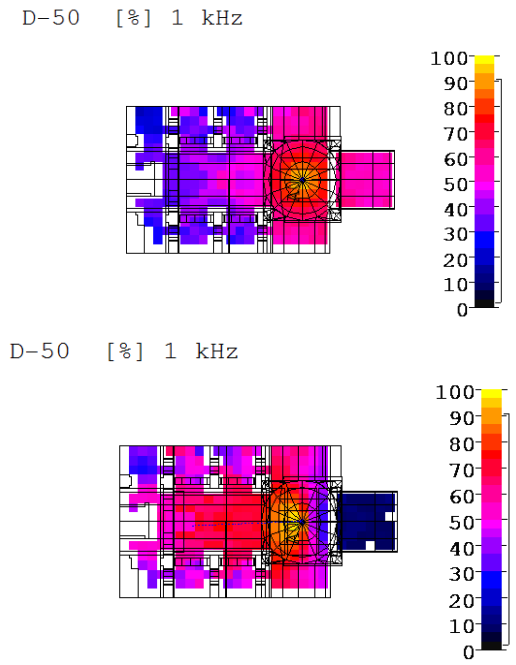


Figure 9. *Definition Index D50 distribution before (up) and after (down) the proposal at 1000 Hz.*

In Figure 9, a clear improvement in D50 values is observable. Generally, the best results for this index in the proposal of rehabilitation are shown in the receivers situated in the central part of the stalls. In the distribution, it is possible to observe that for the 1000Hz band, the largest part of the stalls present values that lower than 93%, while after the restoration and acoustic fitting-out proposal are not lower than 96%. Therefore, the sound of a musical performance (orchestra o chamber group) should be clearer and purer, so that each instrument is clearly distinguished. In this way, a listener could be able to distinguish separately all the notes in a quick musical passage played by the orchestra.

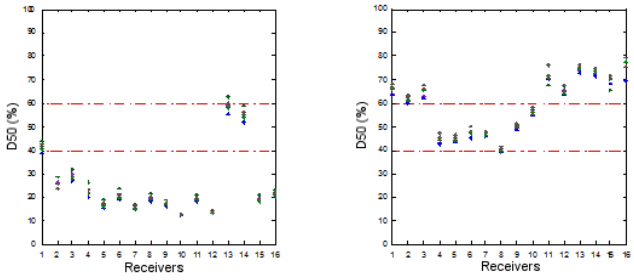


Figure 10. *Definition Index D50 distribution before and after the proposal.*

In Figure 10, the values of the definition index are shown for every receiver and for the frequencies in octave bands from 125 Hz to 4000Hz. It can be observed that before the proposal D50 values are lower than the recommended range (40% to 60 %) [4], for every receiver and every frequency. This is not the case for the receivers located on the stage, although in the case of the simulation after the proposal the D50 values are even higher in most cases.

5.4 Clarity C80 (dB)

The clarity index gives information about the improvement produced after the restoration and acoustic fitting-out in the general proposal. For a room like the Saint Agustine Church, the clarity parameter/limit would have ideal values from 3.1 and 9.1 dB. Before the proposal, the C80 index values in the 16 receivers are between -3 and +3 dB, whereas after the proposal the C80 values change to values between 0 and 9 dB, which produces an improvement in this parameter [4]. In Figure 12, the index C80 (dB) distribution is shown before and after the proposal for the 1000 Hz band.

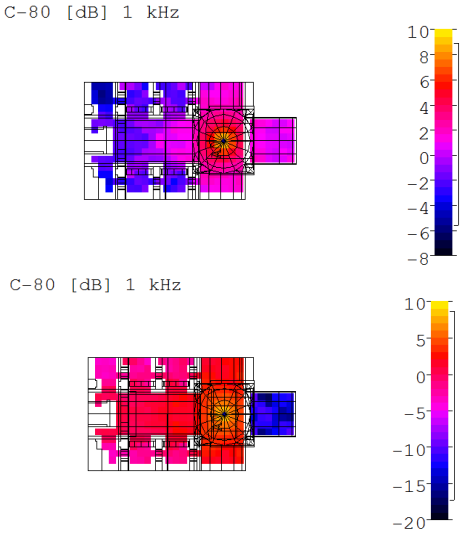


Figure 11. *Clarity index C80 Distribution before and after the proposal.*

Therefore, after applying the proposal in the church, melodies and harmonies (i.e. successive and simultaneous sounds) are perceived separately, allowing the separated audition of sounds produced by every musical instrument.

5.5 Speech Transmission Index (STI)

The STI index confirms a clear improvement in the values obtained after the restoration and acoustic fitting-out proposal. Before the proposal in the 16 receivers the values were between 0.45 and 0.61; then they take values between 0.57 and 0.76. That means that it goes from one poor acoustic qualification of the church to a good acoustic one. In Table II the values obtained for STI index in the 16 receivers can be seen, before and after the general proposal.

The STI index takes a half value of 0.52 before the proposal and after it is 0.65 which represents the clear improvement in the church intelligibility, increasing word clarity/definition, which is now a good acoustic room.

5.6 Rapid Speech Transmission Index (RASTI)

RASTI is devoted to explain the speech transmission for specific frequency bands. Figure 13 shows maps of the RASTI index distribution, before and after the restoration and acoustic fitting-out proposal.

The RASTI index before the restoration and acoustic fitting-out proposal does not exceed the 0.60 value in the stalls, while after the proposal the values range between 0.60 and 0.70. This index goes from sorting the room as a weak or regular to sort as a good one acoustically.

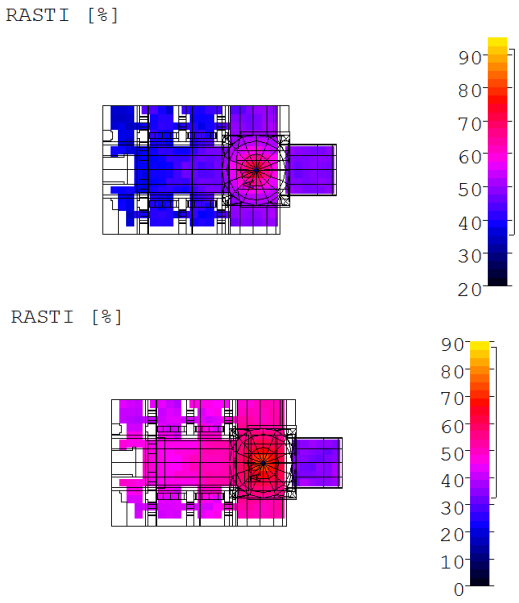


Figure 12. *Clarity index C80 Distribution before and after the proposal.*

6. CONCLUSIONS

The proposed corrective measures proposed to improve the acoustic quality of a room can be basically simplified in two: increasing the room absorption and diminishing the volume. In the global restoration and acoustic fitting-out proposal of Saint Agustine Church both solutions have been used.

On one hand, the church volume has been reduced, in order to diminish the reverberation time, using mobile curtains of velvet, which cover the dome and the arcades of the lateral chapels. Furthermore, establishing curved plates at the end of the stage, the apse area has also been eliminated. In this way, the room volume is diminished, but also shorter and early reflections are obtained in the audience. These reflections try to foster the reflections produced by the lateral walls and the set of sheets on the stage hanged from the dome or from its base. Also, reflections foster the acoustical quality of one room, producing a major sensation of intimacy.

Apart from this, the possible focalizations coming from the curved surfaces as the dome, the apse or the vaults of the chapels are avoided. They can be produced if the source is located near to r/2 (being r radius of the surfaces curvature).

Since in general terms, the part of a room occupied by the performers (usually the stage) should have more reflective areas around it than the part of the audience, the location of the wood plates and the stained-glass sheets which plan the sound towards the outside of the stage, but it also allows the performers to be listened to better. Therefore, a major intimacy sensation is provided to the church, in an acoustic and visual sense.

On the other hand, setting the velvet mobile curtains allows increasing the absorption in the church, by means of the padded stalls.
The proposal has been made in such a way that the actions performed in the church would not cause an irreversible impact on the original building. For this reason, it was always thought to respect the church structure and each solution has a reversible characteristic. This means that any element installed can be removed depending on the use.
The different combinations of elements can be used, depending on the musical style that is going to be performed. Combining the different elements which can vary the acoustic features, they can accomplish the desired conditions for each concert, audition or conference.
In this way, velvet curtains are movable, surfaces at the end of the stage can be supported by wheels and thus they can be moved comfortably and the stained-glass sheets can be removable. These sheets have been made of transparent stained-glass, because in this way it is possible to see the church through them and they are not an obstacle in the visual field. As the stalls seats were already covered, the only thing to change is the upholstery into another more absorbent. Finally, field measures should be taken to check the goodness of the proposal. Therefore this global restoration and acoustic fitting-out proposal could be verified with these measurements in order to validate the new model.

7. REFERENCES

[1] L. CREMER, H. A. MÜLLER: *Principles and Aplications of Room Acoustics*, Applied Science Publishers Ltd. England, 1982

[2] D. MAERCKE, J. VAN MARTÍN: "The prediction of echograms and impulse responses within the Epidaure software", *Applied Acoustics*, vol. 38, pp. 93-114, 1993.

[3] CATT Acoustics v.8. Room Acoustic Prediction and Desktop Auralization. User's manual. 2002. http://www.catt.se

[4] M. RECUERO: *Acondicionamiento acústico*, Paraninfo, 2001.

[5] J. V. GARRIGUES, A. GARCÍA: "Acondicionamiento acústico de la antigua capilla del Colegio Mayor Luis Vives de la Universitat de Valencia", Laboratorio de Acústica. Departamento de Física Aplicada. Universitat de Valencia, 1999.

[6] A. VELA: "Análisis de diferentes métodos de evaluación de la calidad acústica de un local. Aplicación al Teatro Gayarre de Pamplona", Tesis doctoral. Facultad de Física de la Universitat de Valencia, 1996.

[7] M. RECUERO: Acústica arquitectónica aplicada, Paraninfo, 1999.

[8] J. J. SENDRA, T. ZAMARREÑO, J. NAVARRO, J. ALGABA: *El problema de las condiciones acústicas en las iglesias: principios y propuestas para la rehabilitación*, IUCC, ETSA, servicio publicaciones Universidad de Sevilla, 1997.

[9] J. LLINARES, A. LLOPIS, J. SANCHO: *Acústica arquitectónica y urbanística*, E.T.S. de Arquitectura, U.P.V., 1996.

[10] M. RECUERO, C. GIL: *Acústica arquitectónica*, E.U. de Ingenieros Técnicos de Telecomunicaciones, U.P.M., 1993.

[11] F. DAUMAL: *Arquitectura acústica 2. Disseny*, Edicions U.P.C., 2001

[12] J. J. SENDRA, J. NAVARRO: "Proposals of intervention in the acoustical rehabilitation of churches", 11th International *FASE* Symposium, Valencia 15-17 Nov., 1994.

[13] J. LLINARES, A. LLOPIS, J. SANCHO: "Adecuación de la reverberación de una iglesia para su uso como sala de recitales", Proceedings de las Jornadas Nacionales de Acústica, *Tecniacústica*, pp. 219-222. 1993.

[14] A. FARINA, A. COCCHI: "Utilizzo di ex-chiese come sale polifunzionali: la chiesa di S. Lucia a Bologna", Proceedings del *XVIII Convegno Nazionale AIA*, L'Aquila, 18-20 Abril, 1990.

[15] A. FARINA, A. COCCHI: "Correzione acustica di ex-chiese riadattate per utilizzo concertistico: un esempio di progettazione di interventi non Sabiniani con l'ausilio del calcolatore", Proceedings de *Acoustics and Recovery of Spaces for Music*, Ferrara 27-28 Oct., 1993.

[16] D. L. KLEPPER: "The distributed column sound system at Holy Cross Cathedral, Boston, the reconciliation of speech and music", *J. Acoust. Soc. Am.*, vol. 99 (1), pp. 417-425, 1996.

[17] J. SEGURA Y E. A. NAVARRO: "Acondicionamiento acústico del Aula Magna de la Facultad de Farmacia (Burjassot)", Laboratorio de Acústica. Departamento de Física Aplicada.

UNSUPERVISED GENERATION OF PERCUSSION SOUND SEQUENCES FROM A SOUND EXAMPLE

Marco Marchini
Music Technology Group
Universitat Pompeu Fabra
marco.marchini3@gmail.com

Hendrik Purwins
Music Technology Group
Universitat Pompeu Fabra
hendrik.purwins@upf.edu

ABSTRACT

In this paper we present a system that learns rhythmic patterns from drum audio recording and synthesizes music variations from the learnt sequence. The procedure described is completely unsupervised and embodies the transcription of a percussion sequence into a fuzzy multilevel representation. Moreover, a tempo estimation procedure identifying the most regular subsequence is used to guarantee that the metrical structure is preserved in the generated sequence. The final synthesis is performed, recombining the audio material derived from the sample itself. Some examples of generations along with a descriptive evaluation are provided.

1. INTRODUCTION

During the last two decades much effort has been devoted to build computational architectures of musical sequence learning [1, 2]. The result of this research in musical intelligence has often inspired music psychology experiments. One example is the Continuator [3] that has been used to study childhood flow-experience [4]. Moreover, these systems naturally lead to the philosophical question about the nature of "style" in music. The problem has been attacked from many perspectives but the debate among musicologists remains open. Mayer [5] arrives at the conclusion that style is not only a complex concept originating from the interplay of different description levels of a musical piece, but also it is impossible to separate "style" from the social context in which the music has grown.

Many studies have been conducted for the analysis and the generation of music sequences. In particular, in [3], a MIDI-based system for real-time musical interaction was developed, yielding good jazz style music generation.

The handling of memory is a core challenge in music modeling [6]. Whereas the widely used bag-of-features approach neglects any sequential relations between musical events, common n-gram based methods for the representation of musical sequences usually set a maximal fixed length of context. This leads to exponentially growing storage needs to allow the model to account for more complex structures. A solution to this dilemma is offered by the different length Markov model [7]. This model determines the needed context length for each musical sequence individually, therefore maximizing storage economy, without the requirement of excessive storage.

Focusing on the question opened by machine listening systems from an information theory point of view suggests improvements to MIR techniques. In the bag-of-frames approach the distance between two audio signals is independent of the order of the notes. Since most of the musical content generally resides on the temporal organization of the sound material, essential information about the music is lost in the derived descriptors. In fact, the goal of musical intelligence systems is to learn music excerpts in a similar way as the brain does in the first auditory scene analysis process [8]. Thus, these systems indirectly define an operative notion of style.

From a statistical point of view, "style" can be defined as a source of symbols [9]. Equivalently, we can say that, in this context, understanding the style means to find a way to compress [1] the message [10]. Another approach is in the framework of information dynamics (see [11]) in which the analysis of music is related to music cognition. Moreover in [12, 13], causal systems are proposed to capture the emergence of musical categories during the listening process.

Employment of machine learning techniques in generating musical events is crucial to achieve flexibility with respect to different musical contexts. In its architecture, our system is inspired by cognitive principles. In addition, it can be used as a validator for many of the results in music analysis in the way that the quality of the synthesis reveals if the analysis methods use to generate the synthesis have been adequate.

First, we define the system design and the interaction of its parts. Starting from low-level descriptors, we translate them into a "fuzzy score representation", where two sounds can either be discretized yielding the same symbol or yielding different symbols according to which level of interpretation is chosen (Section 2). Then we perform skeleton subsequence extraction and tempo detection to

[1] This means to find a concise representation of the signal without loosing information from the original (lossless data compression). In this way we can store a message (a sequence of symbols) using less bits and then rebuild the original signal by *uncompressing* its shorter version. The complexity of the message turns out to be the key concept that determines the compression ratio (the ratio between the bits occupied by the original message and the ones occupied by the compressed message). The Lempel-Ziv algorithm is an example of such a compressor.

align the score to a grid. At the end, we get a homogeneous sequence in time on which we perform the prediction. For the generation of new sequences we reorder the parts of the score, respecting the statistical properties of the sequence while at the same time maintaining the metrical structure (Section 3). In Section 4, we give a descriptive evaluation of the generated result.

2. UNSUPERVISED SOUND ANALYSIS

The system architecture consists of the following processing stages (cf. Figure 1):

- Segmentation
- Symbolization
  - Feature extraction
  - Feature clustering
  - Sequence structure analysis
  - Temporal alignment
- Generation of audio
  - Adaptive cluster level determination

We will now describe each step of the process in detail.

2.1 Segmentation

First, the audio input signal is analyzed by an onset detector that segments the audio file into a sequence of musical *events*. Each event is characterized by its position in time (onset) and an *audio segment*, the audio signal starting at the onset position and ending at the following contiguous onset. In the further processing, these events will serve two purposes. On one side, the events are stored as an indexed sequence of audio fragments which will be used for the resynthesis in the end. On the other side, these events will be compared with each other to generate a reduced score-like representation of the percussion patterns to base a tempo analysis on (cf. Fig. 1 and Sec. 2.2).

We used the onset detector implemented in the MIR toolbox [14] that is based only on the energy envelope, which proves to be sufficient for our purpose of analyzing percussion sounds.

2.2 Symbolization

We will employ segmentation and clustering in order to transform the audio signal into a discrete sequence of symbols (as shown in Fig. 3), thereby facilitating statistical analysis. However, some considerations should be made.

As we are not restricting the problem to a monophonic percussion sequence, non-trivial problems arise when one wants to translate a sequence of events into a meaningful symbolic sequence. One would like to decide whether or not two sounds have been played by the same percussion instrument (e.g. snare, bass drum, open hi hat...) and, more specifically, if two segments *contain* the same sound in case of polyphony. With a similarity distance we can derive a value representing the similarity between two sounds but when two sounds are played simultaneously a different sound may be created. Thus, a sequence could exist that allows for multiple interpretations since the system is not able to determine whether a segment contains one or more sounds played synchronously. A way to avoid this problem directly and to still get a useful representation is to use a *fuzzy representation* of the sequence. If we listen to each segment very detailedly, every segment may sound different. If we listen very coarsely, they may all sound the same. Only listening with an intermediate level of refinement yields a reasonable differentiation in which we recognize the reoccurrence of particular percussive instruments and on which we can perceive meaningful musical structure. Therefore, we propose to maintain different levels of clustering refinement simultaneously and then select the level on which we encounter the most regular non-trivial patterns. In the sequel, we will pursue an implementation of this idea and describe the process in more detail.

2.2.1 Feature Extraction

We have chosen to define the *salient part* of the event as the first 200 ms after the onset position. This duration value is a compromise between capturing enough information about the attack for representing the sound reliably and still avoiding irrelevant parts at the end of the segment which may be due to pauses or interfering other instruments. In the case that the segment is shorter than 200 ms, we use the entire segment for the extraction of the feature vector. Across the salient part of the event we calculate the Mel Frequency Cepstral Coefficient (MFCC) vector frame-by-frame. Over all MFCCs of the salient event part, we take the weighted mean, weighted by the RMS energy of each frame. The frame rate is 100 frame for second, the FFT size is 512 samples and the window size 256.

2.2.2 Sound Clustering

At this processing stage, each event is characterized by a 13-dimensional vector (and the onset time). Events can thus be seen as points in a 13-dimensional space in which a topology is induced by the Euclidean distance.

We used the *single linkage* algorithm to discover event clusters in this space (cf. [15] for details). This algorithm recursively performs clustering in a bottom-up manner. Points are grouped into clusters. Then clusters are merged with additional points and clusters are merged with clusters into super clusters. The distance between two clusters is defined as the shortest distance between two points, each in a different cluster, yielding a binary tree representation of the point similarities (cf. Fig. 2). The leaf nodes correspond to single events. Each node of the tree occurs at a certain height, representing the distance between the two child nodes. Figure 2 (top) shows an example of a clustering tree of the onset events of a sound sequence.

The height threshold controls the (number of) clusters. Clusters are generated with inter-cluster distances higher than the height threshold. Two thresholds lead to the same cluster configuration if and only if their values are both within the range delimited by the previous lower node and the next upper node in the tree. It is therefore ev-

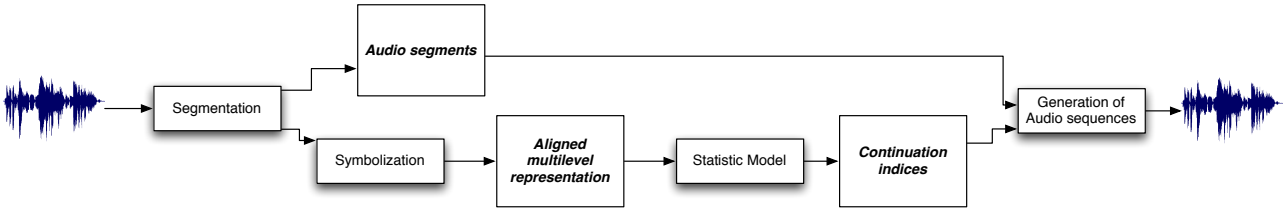


Figure 1. General architecture of the system.

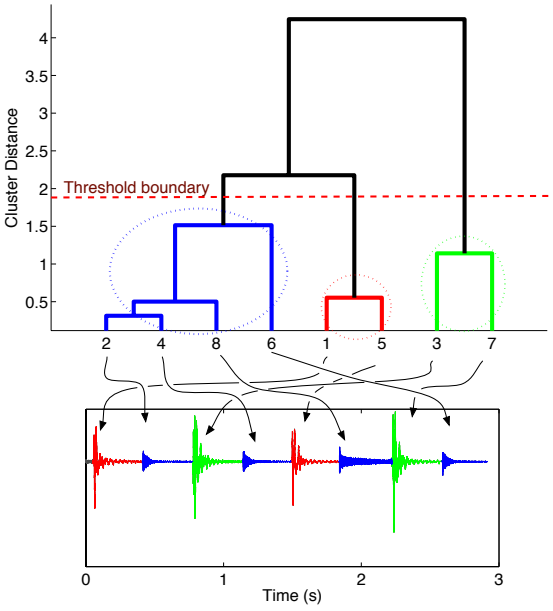


Figure 2. A tree representation of the similarity relationship between events (*top*) of an audio percussion sequence (*bottom*). The threshold value chosen here leads to a particular cluster configuration. Each cluster with more than one instance is indicated by a colored subtree. The events in the audio sequence are marked in the colors of the clusters they belong to. The height of each node is the distance (according to the single linkage criterion) between its two child nodes. Each of the leaf nodes on the bottom of the graph corresponds to an event.

ident that by changing the height threshold, we can get as many different cluster configurations as the number of events we have in the sequence. Each cluster configuration leads to a different symbol alphabet size and therefore to a different symbol sequence representing the original audio file. We will refer to those sequences as *representation levels* or simply *levels*. These levels are implicitly ordered. On the leaf level at the bottom of the tree we find the lowest inter-cluster distances, corresponding to a sequence with each event being encoded by a unique symbol due to weak quantization. On the root level on top of the tree we find the cluster configuration with the highest inter-cluster distances, corresponding to a sequence with all events denoted by the same symbol due to strong quantization. Given a particular level, we will refer to the events denoted by the same symbol as the *instances of that symbol*. We do not consider the implicit inheritance relationships between symbols of different levels.

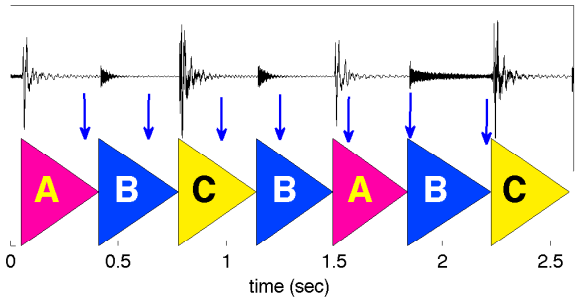


Figure 3. A continuous audio signal (*top*) is discretized via clustering yielding a sequence of symbols (*bottom*). The numbers inside the colored triangles denote the cluster index of the event, related to the type of sound, i.e. bass drum, hi-hat, or snare.

2.3 Level Selection

Handling different representations of the same audio file in parallel enables the system to make predictions based on fine or coarse context structure, depending on the situation. As explained in the previous section, if the sequence contains n events the number of total possible distinct levels is n (see Fig. 4). As the number of events increases, it is particularly costly to use all this levels together because the number of levels also increases linearly with the number of onsets. Moreover, as it will be clearer later, this representation will lead to over-fitted predictions of new events.

This observation leads to the necessity to only select a few levels that can be considered representative of the sequence in terms of structural regularity.

Given a particular level, let us consider a symbol σ having at least four instances but not more than 60% of the total number of events and let us call such a symbol an *appropriate symbol*. The instances of σ define a subsequence of all the events that is supposedly made of more or less similar sounds according to the degree of refinement of the level. Let us just consider the sequence of onsets given by this subsequence. This sequence can be seen as a set of points on a time line. We are interested to quantify the degree of temporal regularity of those onsets.

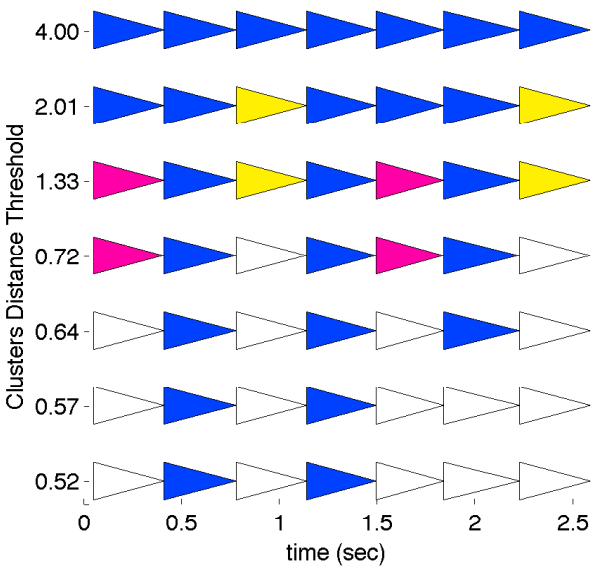


Figure 4. A sequence is displayed in a multi-level representation. Each color represents a unique symbol, a non-trivial cluster, whereas the singletons not belonging to a non-trivial cluster are drawn in white. Note how the number of different colors increases from top to bottom, indicating that the sounds are represented in greater refinement by a larger number of clusters.

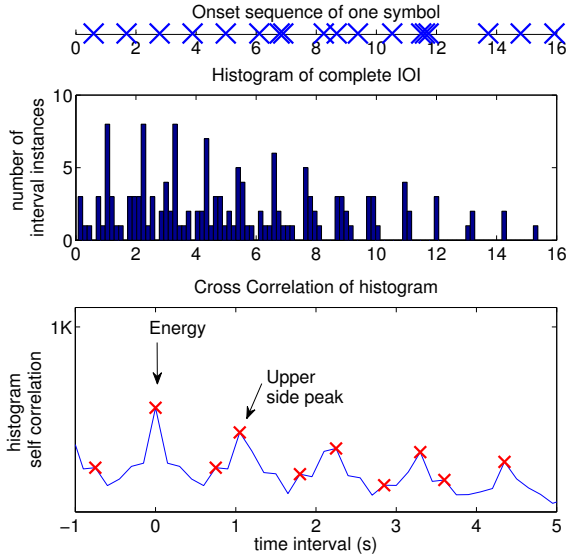


Figure 5. The procedure applied for computing the regularity value of an onset sequence (*top*) is outlined. *Middle:* the histogram of the complete IOI between onsets. *Bottom:* the autocorrelation of the histogram is shown for a subrange of IOI with relevant peaks marked.

Firstly, we compute the histogram [2] of the time differences (CIOIH) between all possible combinations of two onsets (*middle* Fig. 5). What we obtain is a sort of harmonic series of peaks that are more or less prominent according to the self-similarity of the sequence on different scales. Secondly, we compute the autocorrelation $\text{ac}(t)$ (where t is the time in seconds) of the CIOIH which, in case of a regular sequence, has peaks at multiples of its tempo. Let t_{usp} be the positive time value corresponding to its upper side peak. Given the sequence of m onsets $x = (x_1, \ldots, x_m)$ we define the *regularity* of the sequence of onsets x to be:

$$\text{Regularity}(x) = \frac{\text{ac}(t_{usp})}{\frac{1}{t_{usp}} \int_0^{t_{usp}} \text{ac}(t)dt} \log(m)$$

This definition was motivated by the observation that the higher this value the more equally the onsets are spaced in time. The logarithm of the number of onsets was multiplied by the ratio to give more importance to symbols with more instances.

Then we extended, for each level, the regularity concept to an overall *regularity of the level*. This simply corresponds to the mean of the regularities for all the appropriate symbols of the level. The regularity of the level is defined to be zero in case there is no appropriate symbol.

After the regularity value has been computed for each level, we yield the level where the maximum regularity is reached. The resulting level will be referred so as the *regular level*.

We also decided to keep the levels where we have a local maximum because they generally refer to the levels where a partially regular interpretation of the sequence is achieved. In the case where consecutive levels of a sequence share the same regularity only the higher one is kept. Figure 6 shows the regularity of the sequence for different levels.

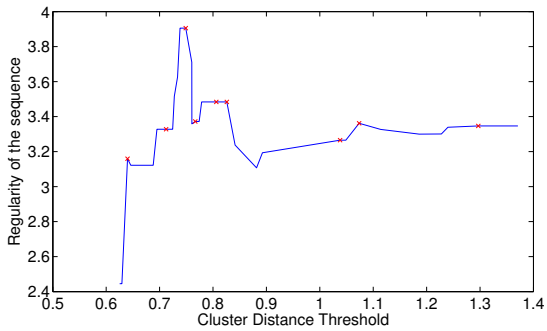


Figure 6. Sequence regularity for a range of cluster distance thresholds (x-axis). An ENST audio excerpt was used for the analysis. The regularity reaches its maximum value in a central position. Towards the right, regularity increases and then remains constant. The selected peaks are marked with red crosses implying a list of cluster distance threshold values.

2.4 Beat detection

In order to predict future events without breaking the metrical structure we use a tempo detection method and introduce a way to align onsets to a metrical grid.

Our starting point is the regular level that has been found with the procedure explained in the previous subsection. On this level we select the appropriate symbol with

[2] We used a discretization of 100 ms for the onset bars.

the highest regularity value. The subsequence that carries this symbol will be referred to as the skeleton subsequence since it is like an anchor structure to which we relate our metrical interpretation of the sequence.

2.4.1 Tempo Detection (Inter Beat Interval)

Once the skeleton subsequence is found, the inter beat interval is estimated with the procedure explained in [16]. The tempo is detected considering the intervals between all onset pairs of the sequence using a score voting criterion. This method tends to give higher scores to the intervals that have more instances and that share many integer ratios with other intervals.

Then the skeleton subsequence onsets are parsed in order to detect a possible alignment of the grid to the sequence. A tolerance of 6% the duration of the inter beat interval is allowed for the alignment of an onset to the grid position. We chose the interpretation that aligns the highest number of instances to grid. After discarding the onsets that are not aligned we obtain a preliminary skeleton grid. In Fig. 7 the procedure is visually explicated.

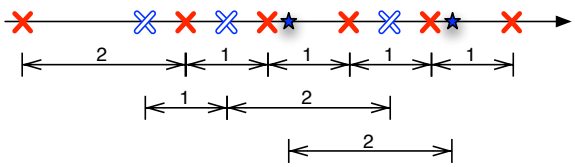


Figure 7. A skeleton sequence is represented in a timeline. *Below*, some possible alignments of the sequences are given based on the measure duration provided by the Dixon method. Each phase interpretation catches some onsets (represented with its own graphical marker) and discards some others. The phase that allows to catch more onsets (the filled red crosses) is selected and the remaining onset are removed from the skeleton grid.

2.4.2 Creation of Skeleton Grid

The preliminary skeleton grid is a sequence of onsets spaced in multiples of a constant time interval. But, as shown in the case of Fig. 7, it can still have some gaps (due to missing onsets). The missing onsets are, thus, detected and, in a first attempt, the system tries to align the missing onsets with one of the onsets of the entire event sequence (not only from the onsets of a certain symbol). A tolerance value of 6% determines whether there is no onset to be aligned and, in this case, the system creates a grid bar in the expected beat position.

At the end of this completion procedure, we obtain a *skeleton grid* that will be considered to be a sequence of beats or, more generally, a sequence of events sharing the same metrical position (the same phase).

Because of the tolerance used for building such a grid it could be noticed that sometimes the effective measure duration could be slightly longer or slightly shorter. This fulfills the idea that the grid should be elastic in the sense that, up to a certain degree, it adapts to the timing of the actual sequence.

The skeleton grid catches a part of the complete list of onsets, but we would like to built a grid where most of the onsets are aligned. Thereafter, starting from the skeleton grid, the intermediate point between every two subsequent beats is found and aligned with an onset (if it exists in a tolerance region otherwise a place-holding onset is added). The procedure is recursively repeated until at least 80% of the onsets are aligned to a grid position or the number of created onsets exceeds the number of total onsets.

In Fig. 8, an example is presented along with the resulting grid where the skeleton grid, its aligned, and the non-aligned subdivisions are indicated by different line markers.

Note that, for the sake of simplicity, our approach assumes that the metrical structure is binary. This causes the sequence to be eventually split erroneously. However, we will see in a ternary tempo example that this is not a limiting factor for the generation because the statistical representation somehow compensates for it even if less variable generations are achieved. A more general approach could be implemented with little modifications.

The final grid is made of *blocks* of time of almost equal duration that can contain none, one, or more onset events. It is important that the sequence given to the statistical model is almost homogeneous in time so that a certain number of blocks corresponds to a defined time duration.

We used the following rules to assign a symbol to a block (cf. Fig 8):

- blocks starting on an aligned onset are denoted by the symbol of the aligned onset,
- blocks starting on a non-aligned grid position are denoted by the symbol of the previous block.

Finally, a phase value is assigned to each block describing the number of grid positions passed after the last beat position (corresponding to the metrical position of the block). For each representation level the new representation of the sequence will be the Cartesian product of the instrument symbol and the phase.

3. STATISTICAL MODEL LEARNING

Now we statistically analyze the structure of the symbol sequence obtained in the last section.

We employ variable length Markov chains (VLMC) for the statistical analysis of the sequences. In [7, 17], a general method for inferencing long sequences is described. For faster computation, we use a simplified implementation as described in [3]. We construct a suffix tree for each level based on the sequence of that level. Each node of the tree represents a specific context that had occurred in the past. In addition, each node carries a list of continuation indices corresponding to block indices matching the *context*.

For audio, a different approach has been applied in [18]. This method does not require an event-wise symbolic representation since it employs the factor oracle algorithm. VLMC has not been applied to audio before, because of the absence of an event-wise symbolic representation we presented above.

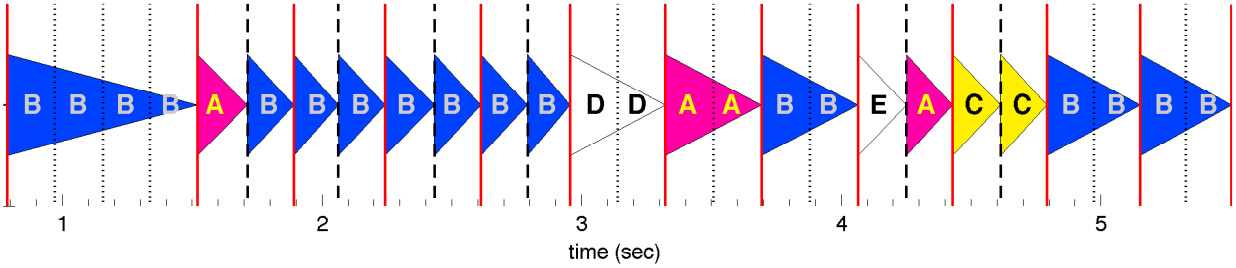


Figure 8. The event sequence derived from a segmentation by onset detection is indicated by triangles. The vertical lines show the division of the sequence into blocks of homogeneous tempo. The red solid lines represent the beat position (as obtained by the skeleton subsequence). The other black lines (either dashed if aligned to a detected onset or dotted if no close onset is found) represent the subdivisions of the measure into four blocks.

3.1 Generation Strategies

If we fix a particular level the continuation indices are drawn according to a posterior probability distribution determined by the longest context found. But which level should be chosen? Depending on the sequence, it could be better to do predictions based either on a coarse or a fine level but it is not clear which one should be preferred. First, we selected the lower level at which a context of at least $\hat{l}$ existed (for a predetermined fixed $\hat{l}$, usually $\hat{l}$ equal 3 or 4). This works quite good for many examples. But in some cases a context of that length does not exist and the system often reaches the higher level where too many symbols are provided inducing too random generations. On the other side, it occurs very often that the lower level is made of singleton clusters that have only one instance. In this case, a long context is found in the lower level but since a particular symbol sequence only occurs once in the whole original segment the system replicates the audio in the same order as the original. This behavior often leads to the exact reproduction of the original until reaching its end and then a jump at random to another block in the original sequence.

In order to increase recombination of blocks and still provide good continuation we employ some heuristics taking into account multiple levels for the prediction. We set p to be a recombination value between 0 and 1. We also need to preprocess the block sequence to prevent arriving at the end of the sequence without any musically meaningful continuation. For this purpose, before learning the sequence, we remove the last blocks until the remaining sequence ends with a context of at least length two. We make use of the following heuristics to generate the continuation in each step:

- Set a maximal context length $\hat{l}$ and compute the list of indices for each level using the appropriate suffix tree. Store the achieved length of the context for each level.
- Count the number of indices provided by each level. Select only the levels that provide less than 75% the total number of blocks.
- Among these level candidates, select only the ones that have the longest context.
- Merge all the continuation indices across the selected levels and remove the trivial continuation (the next onset).
- In case there is no level providing such a context and the current block is not the last, use the next block as a continuation.
- Otherwise, decide randomly with probability p whether to select the next block or rather to generate the actual continuation by selecting randomly between the merged indices.

4. EVALUATION OF EXAMPLES

As a descriptive evaluation, we asked a professional percussionist to judge several examples of continuations as if they were performances of a student. Moreover, we asked him to record two beat boxing excerpts trying to push the system to the limits of complexity and to assess critically the sequences that the system had generated from these recordings. The examples are available on the web site [19] along with some graphical animations explaining the analysis process.

Let us briefly explain what can be seen in these animations. In each video, we see the original sound fragment and the generation derived from it. Each video shows an animated graphical representation where each block is represented by a triangle. The horizontal axis corresponds to the time in seconds and the vertical axis to the clustering quantization resolution. In the beginning, the original sound is played and the animation shows the discovered block representation. At each moment, the currently played block is represented by an increased colored triangle and highlighted by a vertical dashed black line. The other colored triangles highlight all the blocks from the starting point of the measure to the current block. In the second sequence, only the skeleton subsequence is played. In the last sequence, the generation is shown. The colored triangles represent the current block and the current context. The size of the colored triangles decreases monotonically from the current block backwards displaying the past time window considered by the system. The colored triangles are represented only on the levels selected by the

generation strategy. The colors correspond to a symbol in a one-to-one manner.

Four examples were taken from the ENST database (see [20]), one from FreeSound.org and two examples were recorded with the percussionist. From the ENST database, we have selected medium/high complexity examples that we numbered according to our collection list. They are Examples no. 15, 21, 28, and 31 corresponding to the following files of the ENST database:

```
053_phrase_afro_complex_slow_sticks
072_phrase_shuffle-blues_complex_slow_sticks
079_phrase_hard-rock_complex_medium_sticks
088_phrase_waltz_simple_medium_brushes
```

From FreeSound we have selected the popular "Amen Break" loop, because of its common use and manipulations during improvisation sets.

Starting from the latter, according to the percussionist,

> ≪As the starting material is relatively rich the continuation is very good considering the length of the original. Especially interesting is the small looping part in 0.58s. It is very similar to what I would do as a percussionist≫.

It is possible to note how the metrical structure is preserved in those examples due to the introduced tempo restrictions. Moreover, an important feature is that it creates relatively original variations given the short length of the learned examples. Example 28 is commented by the percussionist in the following way:

> ≪Very good. Meter is kept perfectly, and the "drum fills" are provided in appropriate times. A problem is that all fills are played as they appear in the song, and not extended or slightly more complex≫.

For Example 31, the percussionist expressed surprise for the realism of the generation and he referred to Example 15, saying:

> ≪The starting material is very good and rich in this case, so the continuation is rich too. Some fills are expanded and more complex which is very good, although other sequences appear exactly as they did in the original≫.

Referring to the beatboxing examples, he pointed out that the metrical structure is kept correctly but the beats do not vary too much in terms of accent. Then, he added:

> ≪Especially good is the looping of a single beat at 1.20s of the first example, although normally the looping shouldn't be repeated too much to maintain a phrase balance≫.

Finally, as an overall consideration, he pointed out an interesting application of the system:

> ≪If these continuations are used as an accompaniment, they are excellent since they, firstly, maintain a steady rhythm but at the same time evolve and, secondly, they more or less keep the time signature (i.e. strong beats usually land on the strong part of the meter, meters sound conceptually as distinct units)≫.

However, he also mentioned several missing features in comparison to a human percussionist solo.

From our point of view, it is worth mentioning that in Examples 21 and 31 even if the tempo is ternary the generation still preserves the metrical structure. In this case, three ternary beats constitute an event together, causing a bad representation of the sequence (a sort of swing subdivision). The statistical model is still able to select a good block position each time.

The behavior of the system depends on the correct behavior of all its parts. In particular, see [16] for a systematic evaluation of the tempo detection. Nevertheless, some examples show that even when the computed symbolic representation of the audio does not respect directly the underlying musical sequence (e.g. the ternary tempo) the statistic model tends to generate sequences that do not break the metric structure.

5. DISCUSSION

Our system effectively generates sequences respecting the structure and the tempo of the original sound fragment for medium to high complexity rhythmic patterns.

The descriptive evaluation of a professional percussionist confirmed that the metrical structure is correctly managed and that the statistical representation generates musically meaningful sequences. He noticed explicitly that the *drum fills* (short musical passages which help to sustain the listener's attention during a break between the phrases) were handled adequately by the system.

The critics by the percussionist were directed to the lack of dynamics, agogics and musically meaningful long term phrasing which we did not address in our approach.

Part of those feature could be achieved in the future by extending the system to the analysis of non-binary meter. To achieve musically sensible dynamics and agogics (*rallentando, accelerando, rubato...*) of the generated musical continuation for example by extrapolation [21] remains a challenge for future work.

6. ACKNOWLEDGMENTS

Many thanks to Panos Papiotis for his patience during lengthy recording sessions and for providing us with beat boxing examples, the evaluation feedback, and inspiring comments. Thanks to Ricard Marxer for his helpful support. The first author (MM) expresses his gratitude to Mirko Degli Esposti and Anna Rita Addessi for their support and for motivating this work. The second author (HP) was supported by a Juan de la Cierva scholarship of the Spanish Ministry of Science and Innovation.

7. REFERENCES

[1] S. Dubnov, G. Assayag, and R. El-Yaniv, "Universal classification applied to musical sequences," in *Pro-*

ceedings of the International Computer Music Conference, pp. 332–340, 1998.

[2] D. Cope, *Virtual Music: Computer Synthesis of Musical Style*. Cambridge, Massachusetts: MIT Press, 2004.

[3] F. Pachet, "The continuator: Musical interaction with style," in *Proceedings of ICMC* (ICMA, ed.), pp. 211–218, ICMA, September 2002.

[4] A. Addessi, L. Ferrari, S. Carlotti, and F. Pachet, "Young children's musical experiences with a flow machine," in *Proceedings of the 9th International Conference on music perception and cognition*, 2006.

[5] L. Meyer, *Style and music: Theory, history, and ideology*. University of Chicago Press, 1996.

[6] H. Purwins, M. Grachten, P. Herrera, A. Hazan, R. Marxer, and X. Serra, "Computational models of music perception and cognition II: Domain-specific music processing," *Physics of Life Reviews*, vol. 5, pp. 169–182, 2008.

[7] P. Buhlmann and A. J. Wyner, "Variable length markov chains," *Annals of Statistics*, vol. 27, pp. 480–513, 1999.

[8] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. The MIT Press, June 1996.

[9] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, January 2001.

[10] M. J. Weinberger, J. J. Rissanen, and M. Feder, "A universal finite memory source.," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 643–652, 1995.

[11] S. Abdallah and M. Plumbley, "Information dynamics: patterns of expectation and surprise in the perception of music," *Connect. Sci*, vol. 21, no. 2-3, pp. 89–117, 2009.

[12] A. Hazan, R. Marxer, P. Brossier, H. Purwins, P. Herrera, and X. Serra, "What/when causal expectation modelling applied to audio signals," *Connection Science*, vol. 21, pp. 119 – 143, 2009.

[13] R. Marxer and H. Purwins, "Unsupervised incremental learning and prediction of audio signals," in *Proceedings of 20th International Symposium on Music Acoustics*, 2010.

[14] O. Lartillot, P. Toiviainen, and T. Eerola, "A matlab toolbox for music information retrieval," in *Annual Conference of the German Classification Society*, 2007.

[15] R. Duda, P. Hart, and D. Stork, *Pattern classification*. Citeseer, 2001.

[16] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[17] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: learning probabilistic automata with variable memory length," *Mach. Learn.*, vol. 25, no. 2-3, pp. 117–149, 1996.

[18] S. Dubnov, G. Assayag, and A. Cont, "Audio oracle: A new algorithm for fast learning of audio structures," in *Proceedings of International Computer Music Conference (ICMC)*, pp. 224–228, 2007.

[19] "www.youtube.com/user/audiocontinuation," April 2010.

[20] O. Gillet and G. Richard, "Enst-drums: an extensive audio-visual database for drum signals processing," in *ISMIR*, pp. 156–159, 2006.

[21] H. Purwins, P. Holonowicz, and P. Herrera, "Polynomial extrapolation for prediction of surprise based on loudness - a preliminary study," in *Sound and Music Computing Conference*, (Porto), 2009.

EFFICIENT FINITE DIFFERENCE-BASED SOUND SYNTHESIS USING GPUS

Marc Sosnick **William Hsu**
San Francisco State University
Department of Computer Science
marc@marcsosnick.com whsu@sfsu.edu

ABSTRACT

Finite Difference (FD) methods can be the basis for physics-based music instrument models that generate realistic audio output. However, such methods are compute-intensive; large simulations cannot run in real time on current CPUs. Many current systems now include powerful Graphics Processing Units (GPUs), which are a good fit for FD methods. We describe an implementation of an FD-based simulation of a two-dimensional membrane that runs efficiently on mid-range GPUs; this will form a framework for constructing a variety of realistic software percussion instruments. For selected problem sizes, real-time sound generation was demonstrated on a mid-range test system, with speedups of up to 2.9 over pure CPU execution.

1 INTRODUCTION

Powerful Graphics Processing Units (GPUs) are now common in the standard graphics cards of most desktop and laptop systems. While earlier GPUs are tailored for graphics processing, recent GPUs from companies such as Nvidia (http://www.nvidia.com) have adopted more flexible architectures to support general purpose computing. Software support for non-graphics computing on GPUs has also improved significantly in the last few years, with environments such as Nvidia's Compute Unified Device Architecture (CUDA) [8] and OpenCL [9]. As a result, there has been much development of general computing on GPUs; many of these projects are documented at http://gpgpu.org.

We have been exploring the use of GPUs for real-time sound synthesis. An obvious question is whether GPU memory bandwidth can efficiently support real-time audio. Another question is whether the GPU architecture can reliably operate under the additional constraints of a real time application. Focus should be on compute-intensive and parallelizable synthesis algorithms, to leverage GPU functionality.

One scenario is to implement many copies of relatively low-cost sound synthesis units on the GPU, mix the outputs down to a few channels, and transfer the mix to the CPU. This is useful for environments such as rendering auditory scenes with multiple sources. We have rather different research goals; our target application involves building a responsive instrument based on a compute-intensive synthesis algorithm.

We have implemented a finite difference-based simulation for a two-dimensional membrane (see [1, 7]), which runs in real time on the GPU; the architecture of the GPU is particularly well suited for this type of algorithm. Finite difference methods are well known as an effective approach for sound synthesis; see for example [2, 7]. Such methods can be a framework for constructing a number of complex software percussion instruments; some simple sound samples can be found at http://userwww.sfsu.edu/~whsu/FDGPU. Finite difference-based sound synthesis for large or fine-grained membranes and plates is too expensive to run in real time on CPUs. Previous studies on audio processing using earlier generation GPUs and software have been mixed (see for example [14, 4]). Our results show that it is now feasible to implement such compute-intensive real-time sound synthesis algorithms on GPUs. In general it should be possible to realize many computationally expensive physics-based synthesis models as real-time instruments on portable systems.

Our paper is organized as follows. Section 2 overviews related work on high-performance audio computing. In Section 3, we describe the finite difference synthesis algorithm we worked with, and our implementation using CUDA. We present experimental results and measurements in Section 4. Conclusions are drawn in Section 5.

2 RELATED WORK

The website http://gpgpu.org is a major clearinghouse for information on general purpose computing on GPUs. Relatively few audio-related projects are documented on the site. [14] implemented seven audio DSP algorithms on a GPU. [11] studied waveguide-based room acoustics simulations using GPUs.

GPUs have been used in the real-time rendering of complex auditory scenes with multiple sources. In [3], the GPU is used primarily for computing particle collisions to drive audio events. [15] uses the GPU for calculating modal synthesis-based audio for large numbers of sounding objects. [13] proposed a method for efficient filter implementation on GPUs, and applied it to synthesis of large numbers of sound sources in virtual environments.

Faust [10] is a framework for parallelizing audio applications and plug-ins; it does not currently support GPU computing.

Our target application is a real-time instrument based on a compute-intensive synthesis algorithm, such as a finite difference membrane model. Bilbao has studied extensively the use of finite differencing for sound synthesis; see for example [2]. Since large models based on finite difference methods are too expensive for real-time performance on CPUs, work has been done for example on FPGA-based implementations [7]. Our approach leverages GPUs that are already common on commodity systems, and does not require custom hardware.

3 FINITE DIFFERENCE ALGORITHM

We simulate a membrane using the finite difference (FD) method of approximation of the wave equation with dissipation in two dimensions as derived by Adib [1]. A square membrane is modeled with a horizontal x-y grid of points. The continuous function $u(x, y, t)$ is defined on the spatial x and y, and time t; u is the vertical displacement at the point (x, y) at time t. The wave equation *with dissipation* is given as:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} \tag{1}$$

where η is the viscosity coefficient. Expanding with the truncated second-order Taylor expansion:

$$\begin{aligned} &\frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta l^2} + \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta l^2} \\ &= \frac{u^{n+1}_{i,j} - 2u^n_{i,j} + u^{n-1}_{i,j}}{\Delta t^2} + \eta \frac{u^{n+1}_{i,j} - u^{n-1}_{i,j}}{2\Delta t} \end{aligned} \tag{2}$$

where, since the grid is symmetric, $\Delta l = \Delta x = \Delta y$, and $x = i\Delta x$, $y = j\Delta y$, and $t = n\Delta t$ [7]. Solving for $u^{n+1}_{i,j}$:

$$u^{n+1}_{i,j} = \left[1 + \tfrac{\eta\Delta t}{2}\right]^{-1} \left\{ \begin{aligned} &\rho\left[u^n_{i+1,j} + u^n_{i-1,j} + u^n_{i,j+1} + u^n_{i,j-1} - 4u^n_{i,j}\right] \\ &+2u^n_{i,j} - \left[1 + \tfrac{\eta\Delta t}{2}\right] u^{n-1}_{i,j} \end{aligned} \right\} \tag{3}$$

where, from [5]:

$$\rho = \left(v \cdot \frac{\Delta t}{\Delta x}\right)^2 \tag{4}$$

such that v is velocity of the wave in the medium. For our initial experiments, we treat η and ρ as constants, and used known stable values from Land [5].

In a production system with a variable velocity parameter, it will important to test that the system satisfies the so-called *Courant condition* [1]:

$$|v| \le \frac{\Delta x}{\Delta t} \tag{5}$$

to assure system stability.

We implemented u as three 2-D matrices of single-precision (4-byte) floating point numbers so as to maintain compatibility with Nvidia devices of compute capability 1.2 or lower [8]. We use the leap-frog algorithm to calculate the values at $u^{n+1}_{i,j}$ given the values of $u^{n-1}_{i,j}$ and $u^n_{i,j}$ [1]. Boundary conditions are maintained at each iteration by testing the values of i and j and adjusting $u^n_{i,j}$ appropriately. A scalar gain value is used to either clamp the edge (boundary gain = 0) or allow motion dependent on the adjacent internal grid point times the boundary gain (boundary gain < 1) [5]. Corners are given no special consideration. To obtain different sounds, the values of n (grid size), η, ρ, and boundary gain are manipulated. For example, values of $\eta = 2\text{x}10^{-4}$, $\rho = 0.5$, n = 6, and a boundary gain of 0.75 produces a bell-like tone; values of $\eta = 2\text{x}10^{-4}$, $\rho = 0.5$, n = 16, and a boundary gain of 0 produces a drum like tone. Further examples of this can be found at http://userwww.sfsu.edu/~whsu/FDGPU.

To obtain audio output, the membrane must be excited in some fashion, roughly analogous to striking or plucking the membrane. We use a simple Gaussian impulse to initialize/excite the membrane. $u^{n-1}_{i,j}$ is set to 0, and $u^n_{i,j}$ to a Gaussian impulse, as suggested in [2, 5]. To obtain audio output, a point on the membrane is chosen, and the value for $u^n_{i,j}$ is sampled and scaled at each iteration. For our experiment, the center point of the grid was chosen as the output point.

We coded two implementations of (3), one serial and one parallel. As is typical in real-time synthesis applications, we run the simulation for several time steps and store the generated output samples in the audio output buffer. When the audio output buffer is full, it is handed off to the audio driver for playback. The serial implementation (**Figure 1**), is designed to run on the CPU as in [5, 7]. The outermost loop accumulates output samples in the audio buffer. Then we loop over all the grid points to calculate the elements of the $u^{n+1}_{i,j}$ array. Finally, we update the $u^{n-1}_{i,j}$ and $u^n_{i,j}$ arrays, in preparation for the next time-step. This serial implementation is clearly of $O(\text{n}^2)$.

```
For t=0 to t=output buffer size
  For row = 1 to N
    For col = 1 to N
      Update u^{n+1}_{row,col}
      If row, col is boundary
        Recalculate boundary point
      If row, col is sample point
        Copy u^{n+1}_{row,col} to output buffer
    End for
  End for
  For row = 1 to N
    For col = 1 to N
      u^{n-1}_{row,col} = u^n_{row,col}
      u^n_{row,col} = u^{n+1}_{row,col}
    End for
  End for
End for
End
```

Figure 1. Serial implementation of finite difference membrane simulation.

Our parallel implementation of the finite difference simulation for the GPU (**Figure 2**) is written using Nvidia's Compute Unified Device Architecture (CUDA) extension to C, which allows programmers to take advantage of this architecture. Nvidia's GPU hardware is a SIMT (single instruction multiple threads) architecture using scalable arrays of multithreaded streaming multiprocessors [8]. CUDA divides system hardware into *host* and *device*, where the host is the system (PC desktop or laptop) in which the Nvidia device (or GPU) resides, and the device is the Nvidia GPU on which the parallel program, or *kernel*, executes. The host system first prepares the device and then hands off execution of the kernels to the device. Each kernel is executed on the device in a *thread*, and threads are combined into one, two, or three dimensional *thread blocks*. In a kernel, a thread can obtain its unique *x, y, z* position in the thread block, which is what we use to determine the thread's position when calculating *u*. All threads in a thread block execute simultaneously, but can be synchronized [8].

Memory between the host and device can be independent or integrated with system memory, but in either case are addressed separately on the host and device. On some systems page-locked host memory (called *pinned memory*) can be mapped to the device [8]. Pinned memory simplifies and reduces the overhead of asynchronously transferring results from the device to the host.

In our parallel implementation, each grid point update is mapped to a single thread. A thread determines its position in the grid by finding its 2-D location in the thread block [8]. At each time-step, each thread calculates one update of the $u_{i,j}^{n+1}$ array. As with the serial implementation, each thread checks to see if it is at a boundary; if so, it adjusts the current point. The thread that corresponds to the output point also collects data over multiple time-steps, and updates the output buffer. In order to maintain coherence over time, the threads are synchronized at the points illustrated in **Figure 2**.

```
Calc. row and col from thread index
For t=0 to t=buffer size
  Update u^{n+1}_{row,col}
  If row, col is boundary
    Recalculate boundary point
  Synchronize threads
  If row, col is sample point
    Save u^{n+1}_{row,col} in output buffer
  Synchronize threads
  u^{n-1}_{row,col} = u^{n}_{row,col}
  u^{n}_{row,col} = u^{n+1}_{row,col}
  Synchronize threads
End for
End
```

Figure 2. Parallel implementation of finite difference membrane simulation.

To execute each thread block, the host hands off execution to the device. The simulation runs for several time-steps, and the output buffer is filled with the computation results, after which execution on the device stops. If pinned memory is not supported, the host copies the output buffer to the audio output buffer; otherwise the host passes a pointer to the audio driver using the pinned memory as the audio output buffer. The execution of the thread block is repeated for the duration of the output sound.

We were especially interested in two boundary cases. First, if the output buffer size is small, there will be more calls to execute the grid calculation, creating significant setup overhead. Second, if the grid size is too large, the time that it takes to calculate a grid may push latency past acceptable realtime parameters.

4 EXPERIMENTAL METHOD

4.1 System Configurations

We tested our code on three systems. System 1 was a PC with a 2.5 GHz Intel Core 2 Quad running Ubuntu 9.10 with a 2.6.31-20-generic kernel and an Nvidia GeForce GTX285. System 2 was a Mac Book Air with a 1.86 GHz Intel Core 2 Duo and 2 GB of 1067 MHz DDR 3 RAM running OS 10.5.8 and an integrated Nvidia GeForce 9400M. System 3 was a MacPro with dual 3 GHz Intel Quad-Core Xeon and 5 GB of 667 MHz DDR2 RAM running OS 10.5.8 and an Nvidia GeForce 8800 GT.

These systems represent a good cross-section of available midrange cards. The GTX285 is the most powerful of the three, with 240 CUDA cores running at a Graphics clock of 1.48 GHz. The 8800 GT has 112 CUDA cores running at a Graphics clock of 1.5 GHz. The 9400M is a low-end GPU used mostly in systems with restricted power consumption; it has 16 CUDA cores running at a Graphics clock of 0.80 GHz.

The 9400M and GTX285 both support pinned memory, whereas the 8800GT does not. The 9400M is integrated into the motherboard, whereas the 8800GT and GTX285 both are PCI cards. The 9400M's memory is integrated into system memory, while the 8800GT and GTX285 memory is independent of system memory.

4.2 Software Implementation Details

Our parallel software implementation of the finite difference membrane simulation is written in C++ using Nvidia CUDA (The package is available for download at http://userwww.sfsu.edu/~whsu/FDGPU) We use PortAudio (http://www.portaudio.com) in blocking I/O mode as our cross-platform audio interface.

For both serial and parallel versions, the main loop of the simulation runs for a number of cycles and fills the audio output buffer. Data in the output buffer is then passed on to PortAudio for real-time output or to be stored in a file. On systems with pinned memory, samples generated and stored in the audio output buffer are accessed directly through pinned memory. On systems without pinned memory, data in the output buffer is copied from the device to the host. The PortAudio driver blocks until it has received data [12], thus allowing us to clearly test timing by seeing obvious buffer underrun conditions.

5 EXPERIMENTAL RESULTS

On the three systems we outlined above, we tested the audio output quality for real-time performance, for grid sizes from 15 x 15 to 21 x 21, and audio buffer sizes from 8 to 4096. We discovered that, as expected, the larger the output buffer or the larger the grid size, the better the GPU performed, relative to the CPU on the same system. The predominant problem was jitter [6] caused by buffer underruns. On the GTX285 system, with the parallel implementation on the GPU, we experienced clean output across all grid sizes and audio buffer sizes. However, with the serial CPU code, there was jitter when the grid size was greater than 20 or the buffer size was at 4096 samples or larger. On the 8800GT system, we experienced jitter for both parallel and serial versions, when the buffer size was less than 1024 samples and the grid size at 21 x 21. On the 9400M system, we experienced jitter with both parallel and serial versions, when the buffer size was less than 1024 samples, or the grid size was greater than 17 x 17. On all systems, responsiveness was difficult to evaluate objectively; to fill a buffer of 1024 samples at 44100 Hz, would require approximately 23 ms, which [6] identifies as the threshold for perception of latency. It appears that our parallel finite difference simulation, running on the GTX285 system, can be the basis for a responsive software instrument.

While it is difficult to compare performance on the three systems with different CPUs and GPUs, we set up some simple timing experiments to estimate the efficiency of our parallel implementation. We simulated playing a sample for one second, and repeated this five times. We used the built-in CUDA timers to measure the amount of time it took to calculate the samples and transfer the samples from the device to the system, using pinned memory on systems where that is available, and asynchronous transfers for the system without pinned memory. We made measurements for several audio output buffer sizes, and several grid sizes.

System	Buffer Size (Samples)	GPU Time (ms)	Memory Transfer (ms)	GPU Total (ms)	CPU Time (ms)
GTX285	8	1626	0	1626	3060
	512	1062	0	1062	3032
	4096	1067	0	1067	3102
9400M	8	7251	0	7251	4052
	512	5674	0	5674	4088
	4096	2842	0	2842	4133
8800GT	8	2863	705	3568	2562
	512	2095	12	2106	2518
	4096	2110	2	2112	2539

Table 1. Results for fixed 21 x 21 grid and varying output buffer size.

The results of the tests run on our three test systems, with a fixed grid size of 21 x 21 and varying buffer sizes, are summarized in **Table 1**. *Buffer Size* is the size of the output buffer in samples. *GPU Time* is the total execution time in milliseconds of the kernels on the GPU. *Memory Transfer* is the total time in milliseconds to transfer the output buffer from the device to the host; a memory transfer value of 0 indicates that the device supported pinned memory. *CPU Time* is the total execution time in milliseconds of the serial implementation on the CPU. All timings represent a total time over 5 runs of 1- second output each (i.e. total of 220500 samples).

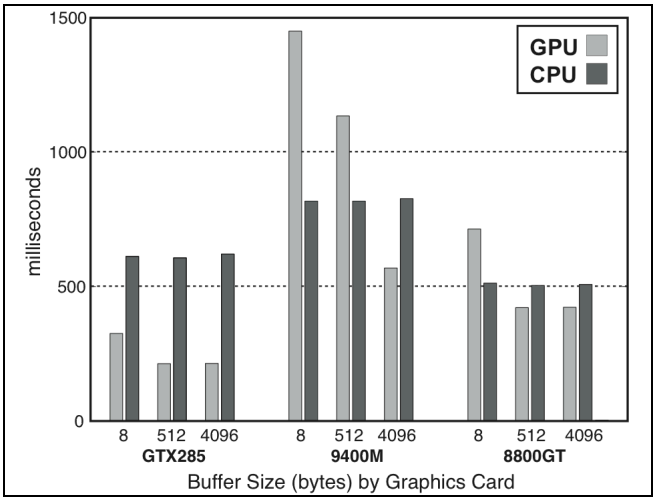


Figure 3. Execution speed with a constant grid size of 21 x 21 points, and varying output buffer sizes.

As can be seen in **Figure 3**, performance on the CPU remains almost constant for all buffer sizes. As the output buffer size increases, generating the same number of output samples requires fewer kernel calls and memory transfers on the GPU; thus the overhead decreases. For the GTX285 system, the performance of the parallel version increased significantly when buffer size increased from 8 to 512, and stayed about constant for larger buffer sizes. The parallel implementation ran faster than the serial implementation, with speedups of 1.2 to 2.9. The 9400M system had the lowest performance of the three. The performance of the parallel implementation increased steadily with larger buffer sizes. For the 8800GT system (no pinned memory), as the buffer size increased, the

System	Grid Size (Points)	GPU Time (ms)	Memory Transfer (ms)	GPU Total (ms)	CPU Time (ms)
GTX285	15 x 15	924	0	924	1577
	18 x 18	984	0	984	2224
	21 x 21	1067	0	1067	3102
9400M	15 x 15	2222	0	2222	1984
	18 x 18	2957	0	2957	3040
	21 x 21	2842	0	2842	4133
8800GT	15 x 15	1411	2	1413	1266
	18 x 18	1743	3	1746	1843
	21 x 21	2110	2	2112	2539

Table 2. Results for a fixed buffer size of 4096 samples, and varying grid size.

overhead for memory transfers decreased as a percentage of total execution time. The parallel code was faster than the serial code only with a buffer size of 512 or greater.

Table 2 summarizes timing estimates with a fixed buffer size of 4096 samples, but with varying grid sizes of 15 x 15, 18 x 18, and 21 x 21. (We were unable to work with larger grid sizes because of GPU memory limitations for our current implementation.)

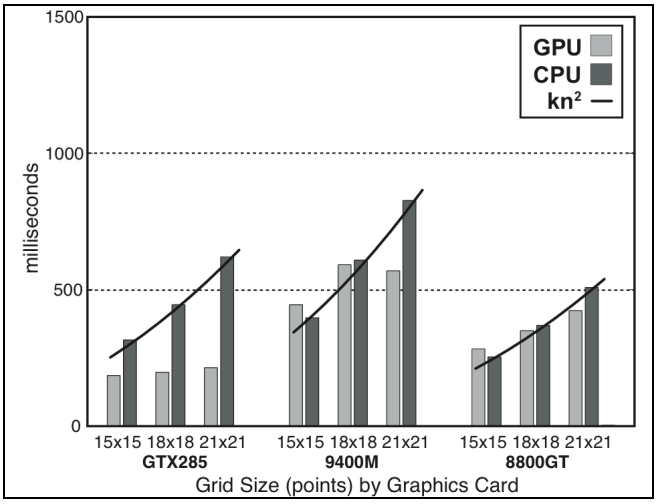


Figure 4. Execution speed with a constant buffer size 4096-samples, and varying grid sizes. For the GTX285, k=0.755; for the 9400M k=1.0; for the 8800GT k=0.629.

As with the previous test, the parallel implementation was faster than the serial on the GTX285 system for all tested grid sizes; it can be seen **Figure 4** that timings for the CPU show an approximate $O(n^2)$ increase with grid size, while GPU timings increase significantly more slowly. With all grid sizes, speedup improved with larger grid sizes. For the 9400M system and 8800GT system, the parallel version was faster for grid sizes 18 and 21, but the serial version was faster for a grid size of 15.

6 CONCLUSIONS AND FUTURE WORK

Our goal for this project was to explore the ability of current mid-range GPU cards to support real-time compute-intensive physics-based synthesis algorithms. We have shown that it is possible to use GPUs to generate real-time audio based on finite difference plate/membrane simulations, but that correct choice of output buffer size and simulation grid size are important. Our straightforward implementation of a parallel finite difference algorithm runs efficiently on our first test system with a GTX285; our less powerful test systems will support adequate performance with selected buffer and simulation grid sizes.

From the results with the 8800GT system, we have shown that memory bandwidth is not a major issue, at least for problems similar to our finite difference code. Newer models of GPU cards that support pinned memory largely avoid the overhead of copying results between the GPU and the host CPU. Larger simulation grid sizes can leverage the parallelism of multiple GPU cores, if the data sizes do not exceed the available GPU memory size. The output buffer size can be increased to reduce kernel call and memory transfer overhead, but at the cost of responsiveness.

Future work will focus on creating a modular production-quality synthesis package using the GPU and finite difference methods, for modeling a variety of percussion instruments. Some limitations of the current implementation must be addressed. Our current version supports only relatively small grid sizes. We are working on distributing the parallel kernel across multiple thread blocks, and using texture memory, to allow for larger or denser grids. Our code is written in the proprietary CUDA extension. We are planning on rewriting the GPU software in the industry-standard OpenCL language [9] and testing it across heterogeneous compute platforms.

7 REFERENCES

[1] A. Adib: "Study Notes on Numerical Solutions of the Wave Equation with the Finite Difference Method," *arXiv:physics/0009068v2 [physics.comp-ph]. 4 October 2000*. Downloaded from http://arxiv.org/abs/physics/0009068v2 on April 15, 2010.

[2] S. Bilbao: "A finite difference scheme for plate synthesis," *Proceedings of the International Computer Music Conference*, pp. 119-122, 2005.

[3] K. van den Doel, D. Knott, D. Pai: "Interactive Simulation of Complex Audio-Visual Scenes," *Presence: Teleoperators and Virtual Environments*, Vol. 13, No. 1, pp. 99-111, 2004.

[4] E. Gallo, N. Tsingos: "Efficient 3D Audio Processing on the GPU," *Proceedings of the ACM Workshop on General Purpose Computing on Graphics Processors*, August 2004.

[5] B. Land: "Finite difference drum/chime," From http://instruct1.cit.cornell.edu/courses/ece576/LABS/f2009/lab4.html, 4/15/2010.

[6] N. P. Lago, F. Kon: "The Quest for Low Latency," *Proceedings of the International Computer Music Conference,* pp. 33-36, 2004.

[7] E. Motuk, R. Woods, S. Bilbao, J. McAllister: "Design Methodology for Real-Time FPGA-Based Sound Synthesis," *IEEE Transactions on Signal Processing*, Vol. 55, No. 12, pp. 5833 – 5845, 2007.

[8] *Nvidia CUDA Programming Guide, version 2.3.1. 8/26/2009.* Downloaded 4/21/2010 from http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/Nvidia_CUDA_Programming_Guide_2.3.pdf.

[9] *Nvidia OpenCL Programming Guide, version 2.3. 8/27/2009*. Downloaded 4/21/2010 from http://www.nvidia.com/content/cudazone/download/OpenCL/Nvidia_OpenCL_ProgrammingGuide.pdf

[10] Y. Orlarey, D. Fober, S. Letz: "Parallelization of Audio Applications with Faust," *Proceedings of the SMC 2009 - 6th Sound and Music Computing Conference*, pp. 23-25, 2009.

[11] N. Rober, U. Kaminski, M. Masuch: "Ray Acoustics using Computer Graphics Technology," *Proceedings of DAFx*, 2007.

[12] B. Roche: Blocking Read/Write Functions. From http://www.portaudio.com/trac/wiki/TutorialDir/BlockingReadWrite, 4/21/2010.

[13] F. Trebien, M. Oliveira: "Realistic real-time sound re-synthesis and processing for interactive virtual worlds," *The Visual Computer,* Vol. 25, No. 5-7, 2009.

[14] S. Whalen: "Audio and the Graphics Processing Unit," Technical Report, Downloaded 4/21/2010 from http://www.node99.org/papers/gpuaudio.pdf.

[15] Q. Zhang, L. Ye, Z. Pan, "Physically-Based Sound Synthesis on GPUs," *Entertainment Computing - ICEC 2005, Lecture Notes in Computer Science,* Vol. 3711/2005.

SHORT TERM PITCH MEMORY IN WESTERN vs. OTHER EQUAL TEMPERAMENT TUNING SYSTEMS

Areti Andreopoulou
Music and Audio Research Laboratory
New York University, New York, USA
aa1510@nyu.edu

Morwaread Farbood
Music and Audio Research Laboratory
New York University, New York, USA
mfarbood@nyu.edu

ABSTRACT

This study investigates the use of short-term memory for pitch recognition in a Western (12-tone) vs. a 10-tone equal temperament context. 10 subjects with at least one year of formal music and theory training participated in an experiment that consisted of two identical music listening tests (one per tuning system) in which they were trained to recall a reference tone and count the number of times it recurred in various short monophonic melodies. In the parts of the experiment where subjects used their short-term memory to execute one-to-one comparisons between the given reference tone and the melody tones, the results were equivalent for both tuning modes. On the other hand, when subjects tried to recall the reference tone directly from long-term memory, the results were noticeably better for the Western tuning context.

1. INTRODUCTION

A considerable amount of research has been done over the past decades regarding the way human brains process, store and recall music pitch. It is clear, for example, that the way our brain responds to music is different from the way it handles visual reference. We are very good in treating colors in a discrete scale, easily distinguishing them, naming them, and recalling them from memory, while we seem to experience pitches in music as a continuum [9].

The population can be divided into groups, according to pitch processing capability; those who have absolute pitch abilities, those who have relative and those who have both. *Absolute pitch* is defined as the ability to either identify and name pitch classes of single tones, or to accurately reproduce a given pitch without any reference, while *Relative Pitch* describes the acquired ability by trained musicians to identify or produce musical intervals [9]. Absolute pitch is a very rare skill, occurring in 1 out of 10,000 people and can be divided into two different skills: *pitch memory*, and *pitch labeling* [10].

While pitch labeling is a unique characteristic of AP possessors, pitch memory is widespread among musically trained and untrained individuals. Schellenberg [12] has shown that adults will little musical training can distinguish the original versions of popular tunes from those pitch shifted by one or two semitones. Other experiments have revealed that listeners can not only identify familiar tunes when played back at different pitch transpositions and tempi, but also accurately reproduce them from memory within a two semitone range and an 8% deviation from the original tempo [9].

Extended research been done regarding characteristics of people that demonstrate absolute pitch abilities. Several theories have been developed based on different factors such as ethnicity, inheritance, age at which individuals began studying music, etc. [7]. Pamela Heaton [8] proposed that autistic musical savants are superior in long-term-memory music tasks, because their cognitive style biases processing of information on local rather than on global features.

Diana Deutsch has, over the years, developed an extended study on pitch memory. Some of her very interesting observations concern the variety of factors that can influence subjects' pitch memory accuracy. She has suggested that recognition precision decreases inversely with temporal separation between the reference and the test tone, if longer than 6 seconds. Furthermore, the presence of distractor pitches between the two tones can also affect subjects' performance. This variation can be considered a function of the pitch relationships and intervals between the distractor tones [3, 4, 5].

In this study we are investigating the use of short-term pitch memory in a 12-tone vs. a 10-tone equal temperament tuning system. To our knowledge, there are no other studies that have attempted a cross-comparison of pitch memory in different tuning systems. Yet, we can use an approach similar to the one employed by Deutsch [3], in which two tones that are either the same or up to a whole tone apart are compared

2. EXPERIMENT

The experiment consisted of two separate sessions (one per tuning system). Subjects were asked to recall a reference tone and count the number of times it appeared in various short, monophonic melodies.

2.1 Participants

Ten subjects (8 female and 2 male) between 23 and 39 years old (mean 29.2) successfully completed both of the listening tests. Two of them claimed to have absolute pitch, and two others claimed to have the ability to recognize certain pitches more easily, even without a reference tone. All subjects reported having at least one year of formal musical training on an instrument (minimum 1, maximum 24, mean 12.3 years) and at least one year of formal theory training (minimum 1, maximum 20, mean 7.3 years).

2.2 Stimuli

Two sets of 46 short monophonic melodies (averaging five measures in length) were composed, one for each of the two listening tests. The stimuli were written in various meters and tempi and had different tonal centers. The target tone could appear between one and five times in a single melody. The first set was composed in 12-tone equal temperament (Western tuning) and the second in 10-tone equal temperament. All melodies were rendered using MIDI piano sounds.

2.3 Procedure

The experiment consisted of two pitch-memory listening sessions. All 10 subjects successfully completed both of sessions in random order, and each session took part on separate days. The procedure and the task were identical for both sessions—the only difference was the set of stimuli used (either 10 or 12-tone equal temperament). The subjects were asked to identify and count the number of times a given reference tone was present in each of the short melodic examples they listened to.

Each session consisted of three parts—intro, training, and testing—and lasted for approximately 45 minutes, with a 10-minute break after the training part. The stimuli were presented to the subjects on a Macintosh computer via headphones, while the graphical interface was a stand-alone application programmed in Max 5.

Before the beginning of the experiment, all subjects had to read a detailed description of the task and the purpose of each part of the session. Subjects were then presented with 6 sample melodies in order to give them the opportunity to familiarize themselves with the interface and task, adjust the volume, and further clarify the test goals as needed. The training phase of the session consisted of 70 melodies (35 different melodies each played twice in a random order), each one preceded by the same reference note. For the 12-tone test the reference note was always the middle D (293.66 Hz), while for the 10-tone one the note was slightly higher in pitch (297.08 Hz). Subjects had to indicate how many times the reference tone appeared in the melodies by pressing the corresponding button. They were clearly advised to ignore all octave equivalent tones.

The training phase was followed by a short 10-minute break, during which subjects had the chance to rest and fill in a questionnaire regarding their music background and their experience with the test so far.

The last part of the session was the testing phase. It consisted of 5 short melodies, presented in random order. This time, no reference tone was given before the beginning of each melody. Subjects were asked, once again, to identify and count the number of times that the previous target tone was present in the melodies by recalling it from memory. Once again, the target tone appeared between 1 and 5 times in each example and subjects were expected to ignore any octave equivalencies.

3. RESULTS

3.1 12-tone equal temperament test

As it can be seen in **Figure 1** all subjects performed well above chance in both the training (minimum 27, maximum 70 correction responses, mean 49, [70%]) and the testing (minimum 1, maximum 5 correct responses, mean 3.7) phases of the 12-tone session.

In the training part, two subjects had a perfect score (one of whom reported having absolute pitch) and two more had more than 60 correct responses (87% accuracy). Three more subjects were clustered around the mean, having a score between 44 and 51 correct answers (62.9% and 72.9%), while the rest had a scores varying from 27 to 31 (38.6% to 44.3%).

In the testing phase, 4 out of the 10 subjects had a perfect score, managing to correctly recall from memory the right pitch all 5 times, while 2 other subjects made only a single error. The rest of the subjects scored below the mean. It is worth mentioning that one of the two subjects who had reported having absolute pitch scored below average in both the training and testing parts.

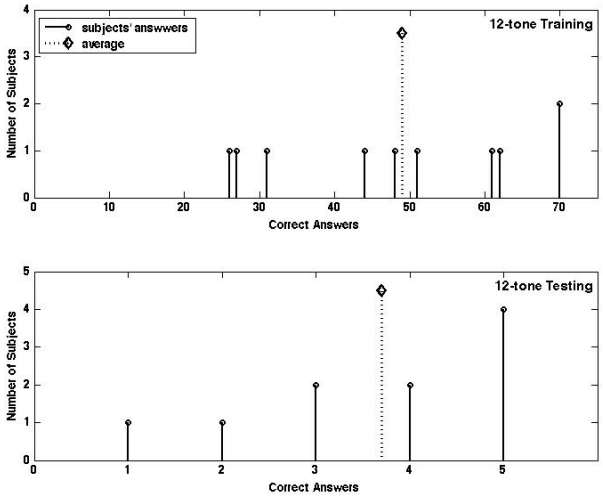


Figure 1. Cross subject evaluation in the 12-tone equal temperament test.

3.2 10-tone equal temperament test

The subjects' performance in the training part of the 10-tone session was nearly equivalent to that of the 12-tone session. The minimum score was 30 correct responses and the maximum 70 (mean 49.3, [70.4%]).

As we can see in **Figure 2**, the top four subjects had a score that ranged between 59 and 69 correct answers (84.3% and 98.6%). Two subject answered 69 out of the 70 questions successfully, one of whom reported having absolute pitch. The next three subjects, clustered around the mean, had scores between 41 and 52 out of 70 correct answers (58.7% and 75.7%), while the bottom three ranged between 30 and 35 (42.9% and 50%).

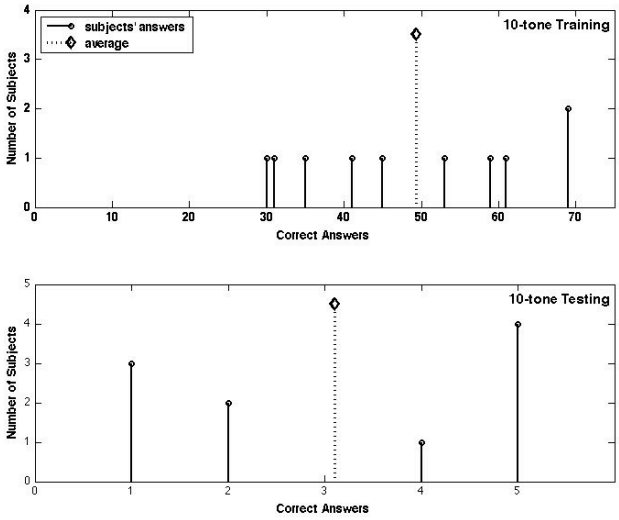


Figure 2. Cross-subject evaluation in the 10-tone equal temperament test.

In the testing phase, on the other hand, we can see differences in the performance of the subjects. Once again 4 subjects had a full score of 5 out of 5, and another had 4 out 5. Yet this time, more subjects scored below the average. As it can be seen in **Figure 2**, 2 subjects had 2 correct answers and 3 managed to recall the correct reference tone from memory only once. One of the subjects who had reported having absolute pitch scored below average in both the training and testing parts of the session.

3.3 Cross-test evaluation

When we initially started conducting the experiment, we expected that the subjects' performance in the 12-tone session would be considerably better than in the 10-tone session due to familiarity with the tonal context. Yet, it quickly became clear that, at least in the training portion of each session, subjects performed equally well in both tuning systems.

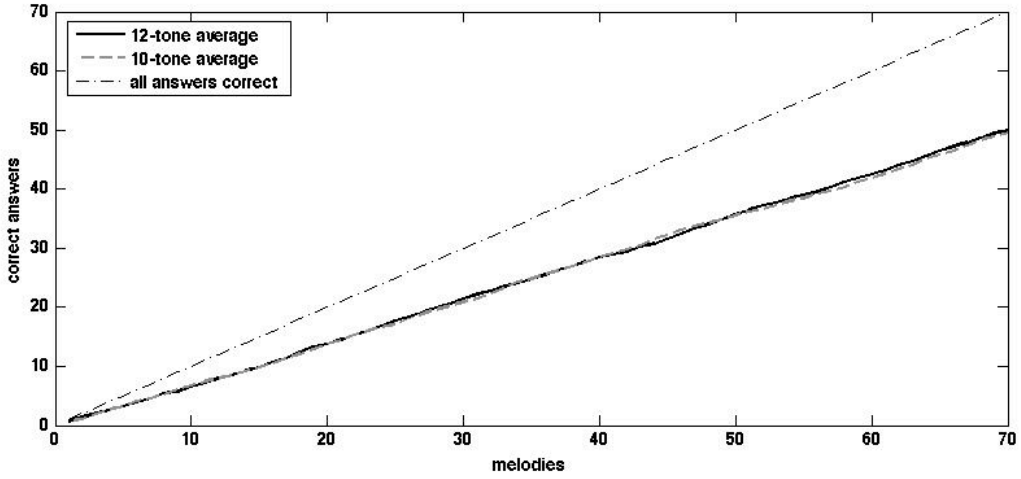


Figure 3. Average cross-session performance of all subjects.

In **Figure 3**, we can see the average progress of all subjects during both of the sessions, versus the perfect score. The horizontal axis corresponds to each new melody the subjects were being tested on during the training part, and the vertical one to the number of correct answers. Each time a subject gives a correct response, the line increases in height; the dashed diagonal, corresponds to perfect score (70/70).

Results show that for each session, the number or correct answers per subject is almost the same for both tuning modes in the training portion. A cross-session evaluation of the training part of the experiment for all subjects is summarized in **Figure 4**.

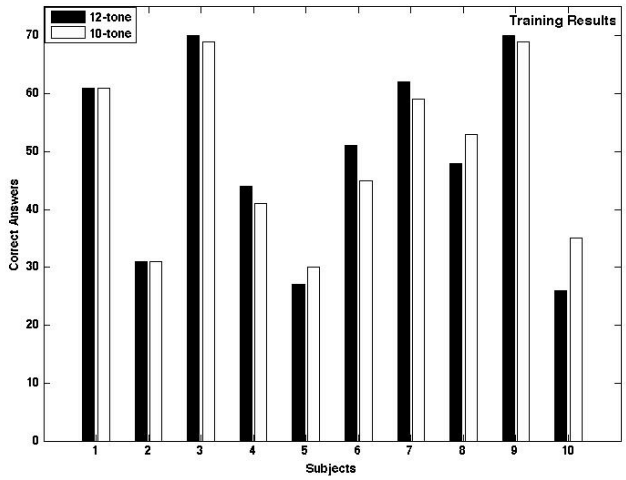


Figure 4. Cross-session performance per subject (Training).

Figure 4 indicates that there is very little difference in performance between the two sessions across all subjects, varying between 0 and 5 out of 70 answers. It may also be noted that while the majority of subjects (7 out of 10) performed a little better or equally well in the 12-tone equal temperament test, the biggest cross-session performance variation can be seen in subject 10, who actually performed better in the 10-tone equal temperament test mode.

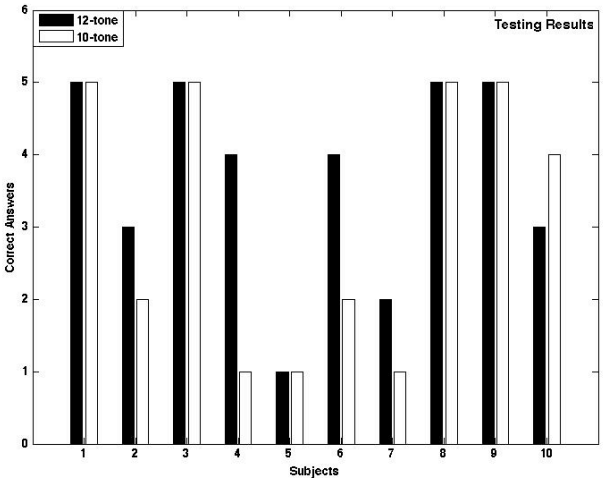


Figure 5. Cross-session performance per subject (testing phase).

The picture is slightly different when we attempt a cross-session performance comparison of the testing phase for all subjects (**Figure 5**). Here the variations in performance are evident (between 0 and 3 out of 5, 0% / 60%). Moreover, half of the subjects have less than 50% accuracy in the 10-tone tuning mode, while that holds true for only two subjects in the 12-tone session. Finally, in 9 out of 10 cases, subjects performed with equal or greater accuracy in the Western tuning testing phase.

4. DISCUSSION

After the completion of the second session, subjects were asked to evaluate the perceived difficulty of the task so far. All but one subject stated that the 12-tone temperament test was equally difficult or easier than the 10-tone one (**Table 1)**.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
12-tone	**8**	**6**	**3**	**6**	**5**	**4**	**4**	**5**	**1**	**3**
10-tone	**8**	**7**	**4**	**7**	**10**	**8**	**7**	**8**	**2**	**2**

Table 1. Evaluation of test difficulty by subjects.

This reaction coincides with our initial hypothesis that subjects would perform better in the 12-tone session due to extended exposure to the tuning system and familiarity with the tonal environment. This is also supported by the overall performance of all subjects in the testing phase of each session, where they had to recall from memory the reference tone that they were trained to identify. Here, as we mentioned before, subjects seemed to more successfully recall the 293.66 Hz reference tone, which corresponds to the middle D (D_4), than the 297.08 Hz tone.

The subjects' results in the testing phase were also in agreement with the information we collected from them, regarding their exposure to non-Western music idioms. Only three subjects responded positively to this question. Yet, even among those three, only one of them performed more accurately in 10-tone equal temperament session (**Figure 5).**

On the contrary, as we have seen in **Figure 4,** the subjects' cross-session performance during the training phase resulted in a highly similar level of accuracy. All subjects were successful in recalling the reference tone. This might be an indication that musically trained listeners can ignore the tonal environment of a certain melody when they are relying on short-term memory to perform one-to-one pitch comparisons between a reference tone, presented immediately before each stimulus and the tones in the following melody. The testing phase, on the other hand, was always preceded by a 10-minute break and no reference tone was explicitly provided. Subjects were thus forced to rely on their long-term memory of the reference tone. It is therefore easy to imagine that the reference tone of the 12-tone test (D_4), had a stronger encoding in the subjects' brains due to extended and repeated exposure to the pitch itself through regular listening of Western music, than that of the 10-tone pitch that was slightly higher in frequency.

It is also worth mentioning the subjects' consistency in the overall perceived difficulty of the experimental task in light of one specific subject's remark. This subject commented, "I find I recall pitch more purely in the microtonal/atonal test. The tonal test introduces interference of relative pitch relationships between notes." It appears that this subject found it easier to recognize the recalled target tone in a one-to-one direct comparison

manner than to establish tonal relationships between the target tone and the key of each test melody.

Three other subjects reported that they were able to develop relative tonal associations between the reference tones and the melodies in the 12-tone equal temperament tests, and felt more insecure in the 10-tone context where such associations were not an option. Finally, one of the two subjects who had reported having absolute pitch mentioned being surprised by the difficulty of the tests. The second one, who had a perfect score in both tests, wrote, "For the Western tuning test, I was able to just count how many times I would hear that D in the melodies. For the second test on the other hand, I was just trying to listen for the target tone and ignore the rest of the notes. After some time, I had memorized it and needed the reference no more." According to that comment, this particular subject had approached pitch comparisons in an absolute rather than a relative manner, in both the Western and the 10-tone tuning modes. Such an approach is fully justified, since, in the case of Western tuning, an absolute perception of frequency is trivial, given the AP listener's learned labels for pitches, while in the 10-tone case, the absence of relative associations make direct comparisons a necessity. These comments imply that a fast and efficient encoding of the reference tone allowed the AP possessor to quickly precede in the 10-tone case much in the same way as in the 12-tone case. It is likely that AP listeners have the ability to create faster, clearer, and stronger mappings of different frequencies that would allow them to continue functioning in an absolute rather than a relative manner. However, this question still remains to be answered.

As indicated by the results of this study, short-term pitch memory can function equally well for both 12-tone and 10-tone equal temperament systems. We would like to further investigate the point at which short-term memory and long-term memory interact or overlap in the case of pitch recall. Further work will include comparing extreme equal-temperament cases, such as 6-tone equal temperament, 12-tone and 16-tone ones, or even unequal temperament scales. We are also interested in expanding the present research to subjects with no formal musical training in order to get an idea of the effect of musical training on subjects' decision-making process in such experiments.

5. REFERENCES

[1] Baddeley, A. *Working memory, thought, and action.* Oxford University Press, 2007.

[2] Chin, Christina S.: "The Development of Absolute Pitch: A Theory Concerning the Roles of Music Training at an Early Developmental Age and Individual Cognitive Style", *Psychology of Music*, Vol. 31, No. 2, 155-171, 2003.

[3] Deutsch, Diana: **"**Mapping of Interactions in the Pitch Memory Store", *Science*, Vol. 175. no. 4025, pp. 1020 – 1022, 1972.

[4] Deutsch, Diana: "Pitch memory: An advantage for the left-handed", *Science,* Vol. 199(4328), 559-560, 1978.

[5] Deutsch, Diana: "The influence of melodic context on pitch recognition judgment", *Perception & Psychophysics,* 31 (5), pp 407-410, 1982.

[6] Gaab, Nadine, Gaser, Christian and Schlaug, Gottfried: "Improvement-related functional plasticity following pitch memory training", *Neuroimage,* Volume 31, Issue 1, Pages 255-263, 2006.

[7] Gregersen, P., Kowalsky, E., Kohn, N. & Marvin, E.: "Absolute Pitch: Prevalence, Ethnic Variation, and Estimation of the Genetic Component", *The American Journal of Human Genetics*, Volume 65, Issue 3, Pages 911-913, 1999.

[8] Heaton, Pamela: "Pitch memory, labeling and disembedding in autism", *Journal of Child Psychology and Psychiatry* 44:4, pp 543–551, 2003.

[9] Levitin, Daniel J:" Absolute memory for musical pitch. Evidence from the production of learned melodies", *Percept Psychophysics*, 56(4): 414-423, 1994.

[10] Levitin, Daniel J. and Rogers, Susan E.: "Absolute pitch perception, coding, and controversies", T*rends in Cognitive Sciences,* Volume 9, Issue 1, 26-33, 2005.

[11] Plantinga, Judy and Traino Laurel J.: "Memory for melody: infants use a relative pitch code", *Cognition*, 98, 2005.

[12] Schellenberg, Glenn E. & Trehub, Sandra E.: "Good Pitch Memory Is Widespread", *Psychological Science*, Vol. 14, No. 3, pp. 262-266. 2003.

[13] Snyder, B. *Music and Memory*. Cambridge, Mass.:MIT Press, 2000.

[14] Trehub, E. Sandra, Schellenberg, E. Glenn & Nakata Takayuki: "Cross-cultural perspectives on pitch memory", *Journal of Experimental Child Psychology.* 100, 40-52, 2008.

AUDIO-BASED MUSIC VISUALIZATION FOR MUSIC STRUCTURE ANALYSIS

Ho-Hsiang Wu and Juan P. Bello
Music and Audio Research Laboratory (MARL), New York University, New York, USA
hhw230,jpbello@nyu.edu

ABSTRACT

We propose an approach to audio-based data-driven music visualization and an experimental design to study if the music visualization can aid listeners in identifying the structure of music. A three stage system is presented including feature extraction, the generation of a recurrence plot and the creation of an arc diagram to visualize the repetitions within a piece. Then subjects are asked to categorize simple forms of classical music with and without audio and visual cues provided. The accuracy and speed are measured. The results show that the visualization can reinforce the identification of musical forms.

1. INTRODUCTION

The detailed study of recorded music is a labor intensive task. This is especially true of the analysis of large audio collections. These collections are difficult to browse given standard catalog and metadata descriptions of music, which provide no information about the musical contents of each recording. Music information retrieval (MIR) research provides a solution to these issues through the development of tools and algorithms that allow for efficient, content-based search and navigation of large music catalogs. In this context, the generation of novel, data-driven and intuitive representations of audio content is necessary to aid the work of musicians and musicologists trying to derive knowledge from these analyses.

Music visualizations have been extensively studied in MIR as ways of helping listeners browse through audio collections and attain a better understanding of their music content [1]. Previous work has mostly concentrated on collection-level visualizations, where tracks are organized according to their similarity or grouped into, e.g., genre or mood categories [2, 3, 4]. Relatively little attention has been paid to the visualization of within-track contents beyond interactive displays based on low-level features [5], or the plotting of feature sequences and intermediate representations (such as self-similarity matrices or other representations used in music structure analysis) that are not necessarily intuitive or informative to non-expert users [6, 4].

In this paper we propose a data-driven approach to the content-based visualization of music structure, based on recurrence analysis of harmonic features. Like previous work on music structure analysis [7, 8, 9], we exploit the existence of patterns of repetition in music. However, we do not assume the common view of music structure as a high-level concatenation of blocks, and thus make no decisions about segmentation boundaries. Instead, we adopt the approach, pioneered in [10] with MIDI data, where data recurrences are visualized by means of arc diagrams, and high-level structure is in the eye of the beholder. The proposed visualization approach is evaluated on its ability to improve the accuracy and speed with which users identify simple forms in classical music.

The remainder of this paper is organized as follows: the details of the visualization approach are described in section 2; the evaluation methodology is discussed in section 3; the results and discussion of the subjective evaluation are presented in section 4; while section 5 presents our conclusions and plans for future work.

2. VISUALIZATION APPROACH

The proposed approach can be subdivided into three main stages: first we extract low-level, harmonic features from the audio signal; second, we project the feature sequence into phase space and compute a recurrence plot from this data; and finally, we generate an arc diagram characterizing the repetitions in the music data stream. The block diagram of the system is shown in Figure 1.

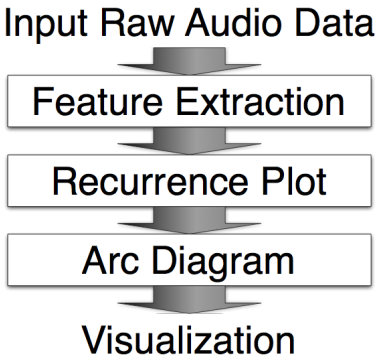


Figure 1. Visualization approach

2.1 Feature Extraction

In our analysis we use chroma features to represent harmonic content in the music signal. Chroma features are commonly obtained from short-term spectral analysis, e.g. via the STFT. For each analysis frame, they represent the signal's energy across the 12 pitch classes of the chromatic scale of western tonal music. The concatenation of these 12-dimensional chroma vectors across time is known as a chromagram.

In our implementation, chroma features are computed via the constant-Q transform [11], a spectral analysis technique in which frequency domain channels are logarithmically spaced, and are thus closely related to the frequency resolution of the human hearing system. First the signal is downsampled to $f_s = 5512.5Hz$. Next, we compute the STFT using a 1024 samples-long Hann window and a hop size of 512. The spectrum is then multiplied by a constant-Q kernel computed with a minimum frequency of 73.42 Hz, a resolution of 36 bins per octave and a 3-octave span. The dimensionality of the chromagram, shown in Figure 2(a), is reduced to 12 bins with a weighted sum across each 3-bin pitch class neighborhood.

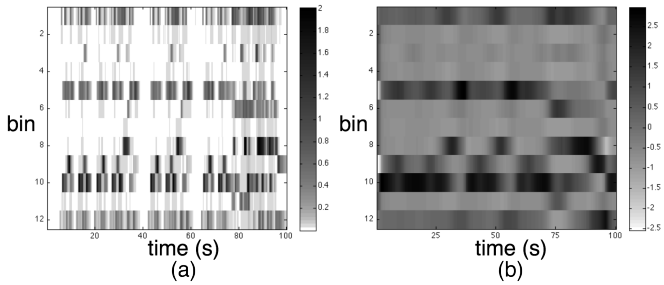


Figure 2. (a) Standard chromagram and (b) chromagram after low-pass filtering, standardization and resampling.

Finally, the features are low-pass filtered, standardized to zero mean and unit variance, and resampled to a resolution of 2 frames per second, as shown in Figure 2 (b).

2.2 Recurrence Plot

A recurrence plot (RP) is a method for analyzing nonlinear dynamic systems [12]. It is visualized as a binary square matrix, in which value ones correspond to pairs of times (indicated by row and column indices) at which a state of the dynamic system recurs. The RP is derived from the so-called phase space, in which all possible states of a system are represented as unique regions. We can consider a chromagram as the output of a nonlinear dynamic system, with each vector corresponding to a trajectory point in a multi-dimensional space. Of course, chromagrams are not produced by dynamic systems, but assuming so allows us to create an intuitive representation able to fully characterize harmonic repetitions in music.

Let us assume a one-dimensional time series $x(t)$, as illustrated in Figure 3(a). We can reconstruct its phase space trajectory using a process known as time-delay embedding. In this process we choose an embedding dimension m, and

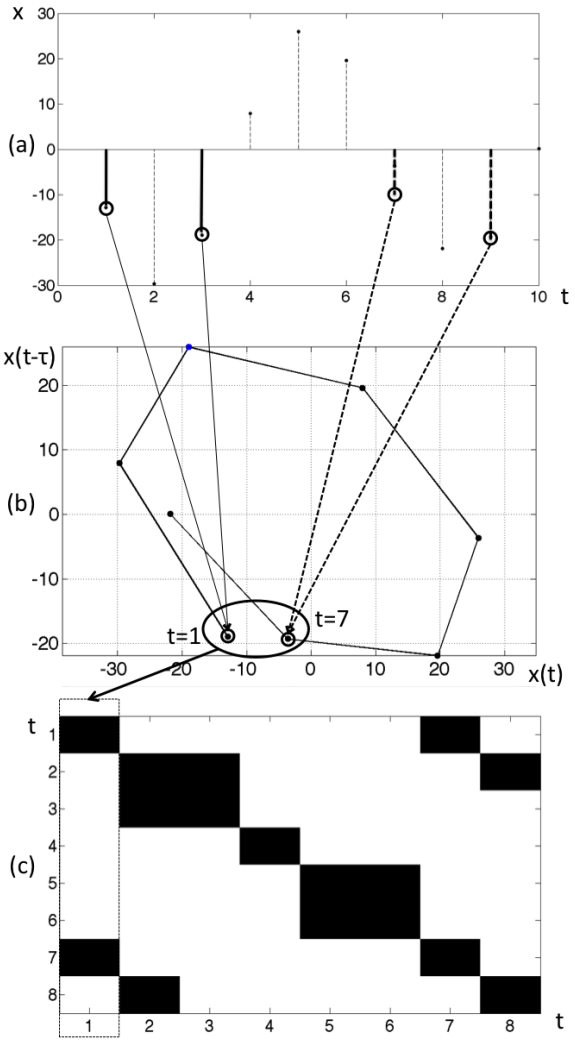


Figure 3. (a) Time series (b) Phase space representation (c) Recurrence plot.

a time delay τ, such that each time t in the series is now represented by a vector obtained by concatenating the values $x(t), x(t-\tau), ..., x(t-(m-1)\tau)$. As can be seen in Figure 3(b), for $m = 2$ and $\tau = 2$, each of these vectors describes a point in the m-dimensional phase space.

Time delay embedding can be also applied to multi-dimensional time series, such as the chromagram $C = \{c_{k,i}\}$, where $i = 1 \ldots N$, the length of the time series, and $k = 1 \ldots 12$, the dimensionality of the chroma vector:

$$\vec{x_c}(i) = (c_{1,i}, c_{1,i-\tau}, \ldots, c_{1,i-(m-1)\tau}, \cdots \\ c_{12,i}, c_{12,i-\tau}, ..., c_{12,i-(m-1)\tau}) \quad (1)$$

We can compute the recurrence plot R for this trajectory, such that $R(i,j) = 1$ if $\vec{x_c}(i)$ and $\vec{x_c}(j)$ are no farther than a distance of ϵ from each other in the phase space, and $R(i,j) = 0$ otherwise. This can be expressed as:

$$R(i,j) = H(\epsilon - \parallel \vec{x_c}(i) - \vec{x_c}(j) \parallel), \\ \vec{x_c}(i) \in \mathbb{R}^m, i, j = m, ..., N, \quad (2)$$

where N is the number of frames in the original time series, ϵ is a threshold value, $\| \cdot \|$ is a norm (e.g. Euclidean norm), and H is the unit step function. Figure 3(c) shows the resulting RP for the phase space in Figure 3(b).

Previous research into applying phase space and recurrence plots to music analysis includes the visualization of expressive timing in piano performance [13] and the state-of-the-art in cover song identification using a variant of RP that can be computed on pairs of songs [14]. Notably, self-similarity matrices, which are widely-used in MIR, are a special case of RP known as *distance* plots [12], computed using $m = \tau = 1$.

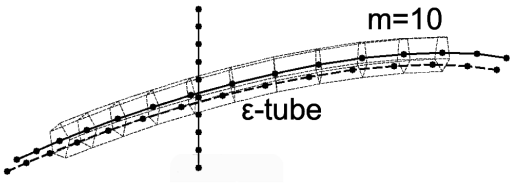


Figure 4. Tangential motion.

Figure 4 serves to illustrate the effect of changing the values of m and τ. The illustration shows a neighborhood of the phase space containing 3 segments of a trajectory: two running in parallel from left to right, and a third one running perpendicular from top to bottom. At the crossover point, when $m = \tau = 1$, it can be seen how all three sections of the trajectory are close enough to be considered recurrences of each other (resulting in values of 1 in the RP). An example RP with $m = 1$ is shown at the top of Figure 5. However, as $(m-1)\tau$ increases, the size of the neighborhood of points that need to be in close proximity to generate a recurrence also increases, weeding out recurrences generated by *tangential* contact. The left column of Figure 5 shows the changes in the RP as m increases. In our visualization experiments, we have heuristically chosen $m = 25$ and $\tau = 1$ as appropriate embedding values.

In both Figure 4 and Figure 5, we can also observe that due to the natural proximity between consecutive points in the trajectory (including between each point to itself), there is an important concentration of activations in the RP along its main diagonal. These activations, however, carry no information about recurrences in the data, and need to be ignored for visualization purposes. A common solution to this problem is to exclude an area of arbitrary width contiguous to the main diagonal by using a *Theiler* window [15]. In our experiments, the Theiler window size is set to 10% of the length of the resampled feature sequence.

Finally, for visualization purposes, it is important to ensure that RPs from different songs contain roughly the same amount of information. In this paper we use recurrence rate (RR), one of several measures commonly used to quantify the properties of RPs [12]. RR measures the density of the plot as a percentage of its recurrences:

$$RR = \frac{1}{N^2} \sum_{i,j=1}^{N} R(i,j) \tag{3}$$

In our implementation RR is used as a threshold on the amount of information we would like to show in the visualization, such that the higher the rate, the denser (and noisier) the plots become. Conversely, the lower the rate, the sparser, and potentially uninformative, they are. After informal experimentation, $RR = 0.2$ was chosen as a good trade-off between both those extremes.

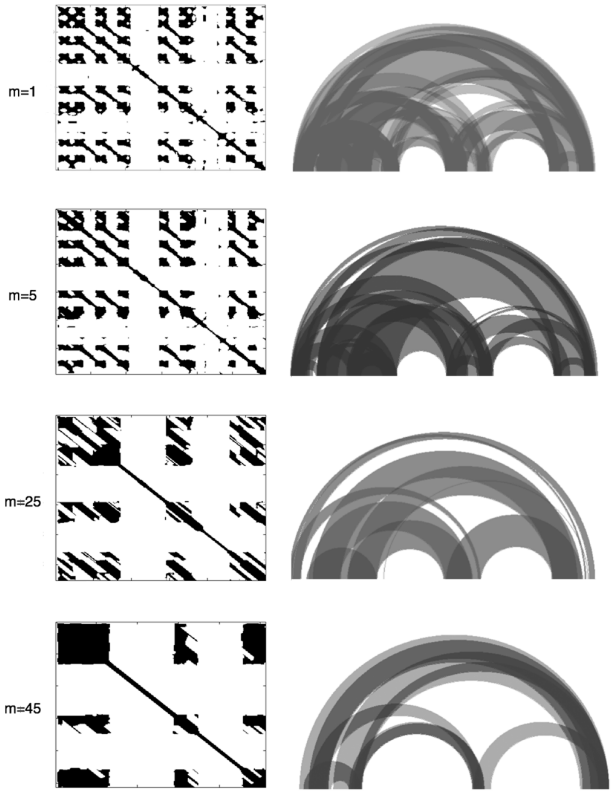


Figure 5. Recurrence plot (left column) and corresponding arc diagram (right column) with different embedding dimension m.

2.3 Arc Diagram

Arc diagrams have been extensively used to represent complex patterns of data recurrence in fields such as biology and physics [10, 16]. They consist of arcs connecting points of repetition in the data stream. Thus, we can simply convert the above recurrence plots into arc diagrams by representing $R_{i,j} = 1$ as an arched line connecting the ith and jth frame of the data sequence. An example diagram can be seen in Figure 6 (a), where the horizontal axis represents discrete time, and $R_{1,9} = R_{2,10} = R_{5,15} = 1$.

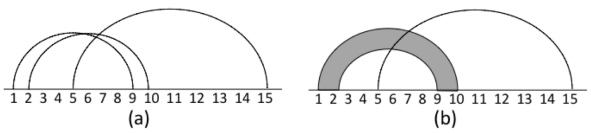


Figure 6. (a) 1 to 1 connection arc diagram (b) Arc diagram with grouping.

To avoid too dense a visualization by repeatedly con-

necting all single arcs, we group neighboring lines into wider arcs. The details of the grouping method are described in Algorithm 1. First we define each group G_s as a vector of four values indexing the arc boundaries, such that $G_s(1)$ and $G_s(2)$ are the left and right boundaries of the leftmost part of the arc, while $G_s(3)$ and $G_s(4)$ are the boundaries of the rightmost part of the arc. Next, we evaluate every point $R_{i,j} = 1$ in the RP, comparing i and j, respectively, with the left and rightmost sets of boundaries of each existing group. If i, j are contiguous to the boundaries of a given group, then they are added to it and the group's boundaries are updated. If i and j do not belong to any existing group, then a new group is created with i and j as the initial boundaries $G_s(i, i, j, j)$. We repeat this process for all points in the RP, until we obtain a final list of groups to be used in the drawing of the arc diagram.

Algorithm 1 Arc diagram grouping method

```
Create a group list G_s(0,0,0,0) with four entries
{G_s(1), G_s(2) leftmost and G_s(3), G_s(4) rightmost part boundaries}

for i ≤ j and i, j = 1 to N do
  if R_{i,j} = 1 then
    for s = 1 to length(G_s) do
      if G_s(3) ≤ j ≤ G_s(4) and 0 ≤ G_s(1) − i ≤ 1 then
        G_s(1) = i
      else if G_s(3) ≤ j ≤ G_s(4) and 0 ≤ i − G_s(2) ≤ 1
      then
        G_s(2) = i
      else if G_s(1) ≤ i ≤ G_s(2) and 0 ≤ G_s(3) − j ≤ 1
      then
        G_s(3) = j
      else if G_s(1) ≤ i ≤ G_s(2) and 0 ≤ j − G_s(4) ≤ 1
      then
        G_s(4) = j
      else
        create new group G_s(i, i, j, j)
      end if
    end for
  end if
end for
```

In the creation of the arc diagram visualization, we have borrowed several implementation strategies from [10]. For example, we use translucent color in order to clearly depict multiple layers of arcs, with color depths (saturation) used to resolve arc overlap in the limited pixel space. Another strategy is to show grouped arcs as a single, wider arc as in Figure 6 (b), rather than as a collection of individual arcs as in Figure 6 (a), where the perceived order of the arc boundaries is reversed. The result of the process can be observed on the right column of Figure 5, which shows a number of examples of arc diagrams for varying values of m. It can be seen how the choice of embedding parameter affects the density and clarity of the visualization.

3. EXPERIMENTAL DESIGN

3.1 Task

We conduct a preliminary experiment to study if these visualizations can help convey information more intuitively and efficiently than simply listening to the music. The task is to identify musical forms as belonging to one of four categories (strophic, binary, ternary and rondo), with and without aid from audio/visual cues.

Four different combinations of visual and audio cues are provided in the experiment in order to compare the influences of each of them on the perception of musical forms. In the first category, both visualization and music are provided. In the second category, only music is provided without any visual cue. In the third category, only visualization is provided without any music playing. In the fourth category, we provide both music and segmentation boundary information obtained using the Echonest API [17].

In order to prevent confusing definitions arising from listeners' different interpretations and understandings of musical form, the four categories to be identified are represented with capital letters indicating the segments which comprise the structure. Strophic form is represented as AA, AAA or AA'. Music in this form contains one main theme which is repeated either with or without slight variations. Binary form is represented as AB or ABAB. Music in this form consists of two main alternating themes, always ending with the second theme. Ternary form is represented as ABC or ABA'. Music in this form has three main themes, or two main themes plus a re-statement of the first theme with or without variations. Rondo form is represented as ABACA, ABACABA or ABACADAEA, where we have a recurring main theme alternating with other different (usually contrasting) themes.

3.2 Methodology

First, the subjects are given descriptions of the task, the explanations of musical forms and the experiment environment. Example tests of each category of audio/visual cue combination are then given to the subjects in order to familiarize them with the system and the experimental process. Next, the subjects are asked to go through forty music pieces with combinations of different forms and visual/audio cues provided. A screen shot of the test environment is shown in Figure 7. Subjects can click on any location on the image they are interested in and listen to the music starting from the corresponding location in the audio track. They can listen to the same piece for as long as they want until they make a decision, at which point they click on one of the check boxes to select a form and move on to the next test. The total length of time they spend on each test is recorded as a measure of the speed of recognition.

The experimental system is implemented using Processing [18]. Processing is a Java-based programming language which provides programmers with quick graphical, visualization and interaction prototyping environment. It also provides certain control abilities for user interface design.

3.3 Subjects

Twenty people were invited to participate in this experiment, ten males and ten females, all over the age of eighteen. Ten of them are trained classical music players and

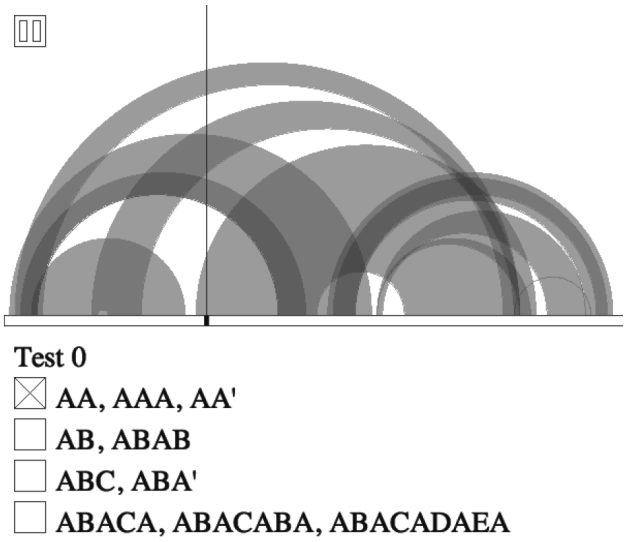


Figure 7. Screen shot of test environment.

ten of them are not. All of them listen to music on a regular basis.

3.4 Test Music

For each of the four forms described above, ten classical music pieces are selected. Strophic form pieces are selected from Lieder of the Classical era. For binary and ternary forms, pieces are chosen from Schumann's "Album for the Young" and Bach's "Notebook for Anna Magdalena Bach". Rondo form pieces are selected according to the examples in the music theory book "The Analysis of Musical Form" [19]. These pieces are mostly from the works of Beethoven and Mozart of the Classical era.

4. RESULTS AND DISCUSSION

The average accuracies and response times for all test are depicted in Table 1. Results are categorized by modes of audio/visual interaction, including a combination of music, arc diagram visualizations and segmentation boundaries.

	Vis/Mus	Mus only	Vis only	Seg/Mus
Accuracy	57.5%	42.5%	37%	45%
Avg time	165s	265s	29s	198s

Table 1. Overall accuracy (%) and average time (seconds) for each category.

The relatively-low levels of classification accuracy illustrate the inherent difficulty of the task, with subjects taking an average of 4 minutes and a half per track to achieve 42.5% accuracy in listening-only tests. It is immediately evident that the use of visual cues (both segmentation boundaries and the arc diagram) improves accuracy and speed. The addition of the proposed visualization is most beneficial, improving accuracy by an average of 15% and reducing analysis time by an average of 100 seconds, significantly better than music-only and music plus segmentation results.

However, visualization-only experiments resulted on worst average accuracy and, unsurprisingly, fastest recognition speed. This indicates that while the arc diagrams help to draw attention to important information in the music signal, they are by themselves not enough to robustly convey information about the musical structure. After tests, subjects informally reported that their preference was to listen to the whole piece before looking for details and finding repetitions. In this context, visualizations helped to guide navigation, but failed to succeed in replacing the music recording as the main information channel.

RE \ GT	Strophic	Binary	Ternary	Rondo
Strophic	52%	14%	8%	17%
Binary	10%	54%	24%	8%
Ternary	11%	17%	56%	7%
Rondo	27%	15%	12%	68%

Table 2. Confusion matrix of the category with both visualization and music.

Table 2 provides a closer look at recognition results using both audio and the proposed visualization. It shows the confusion matrix, with rows representing ground truth values (GT) and columns the subjects' results (RE). Most confusions fall into one of two categories: strophic/rondo confusions and binary/ternary confusions. The former, can be partly attributed to the similarities between the corresponding arc diagrams. Figure 8 shows diagrams for two pieces in strophic (a) and rondo (b) form, respectively. It can be seen how variations of the repeating theme in (a) result on gaps in the visualization that can be easily confused with the representation of an alternating theme, as in the case of (b). This is also partially the result of the density constraints imposed on the diagram, i.e. by definition strophic forms will tend to result in denser diagrams. However, the constraints are necessary to avoid too-noisy or too-sparse diagrams in other styles, and adaptation has proven elusive without prior information about the music. Confusions between binary and ternary forms might be caused by the nuanced and ambiguous difference between AB, ABA and ABA' structures. Thus, more ternary music is miscategorized as binary than the opposite.

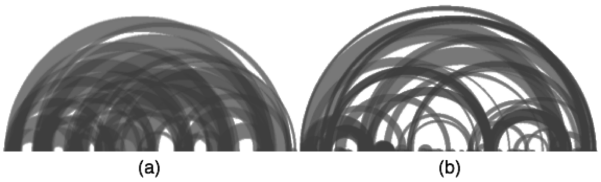


Figure 8. Visualization examples of (a) Strophic form and (b) Rondo form.

5. CONCLUSIONS AND FUTURE WORK

This paper introduces an audio-based, data-driven visualization of music structure, obtained through the use of chroma features, recurrence plots and arc diagrams. Additionally, it presents a preliminary study exploring the ability of the proposed visualizations in helping complex tasks such as music form recognition. Our results indicate that these visualizations can reinforce the identification of musical structures, both reducing the time and increasing the accuracy of the analysis. They provide an efficient way in aiding listeners navigate through music. However, results also indicate that the proposed visualizations are not yet enough, on their own, to convey structural information to users, making them good complements, but not alternative representations, in the analysis of recorded music.

To address these issues, we are currently working on incorporating other musical attributes into the visualization. Previous work has discussed the multi-dimensional nature of musical structure, with important cues provided not only by harmonic and melodic patterns (which the chroma features attempt to characterize), but also by rhythmic and textural characteristics. The data-driven nature of the process means that we can easily compute these visualizations from features such as MFCCs or so-called tempograms [20], which are intended to represent those attributes. However, integrating multiple dimensions into an intuitive diagram is far from trivial and we are actively investigating different color-scheme and layering strategies to solve this problem.

Additionally, we are planning to embrace the multi-scale nature of music, allowing users to interactively navigate across micro and macro readings of the data according to their information needs, e.g. allowing users to zoom into the structure of a song at the phrase-level, to visualize local patterns of recurrence. A long-term goal is to integrate multi-scale analysis at the collection, work and track level.

ACKNOWLEDGEMENTS: This material is based upon work supported by the National Science Foundation (grant IIS-0844654).

6. REFERENCES

[1] J. Donaldson and P. Lamere: "Using Visualizations for Music Discovery," In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR-09)*, Kobe, Japan. October 2009.

[2] E. Pampalk, A. Rauber and D. Merkl: "Content-based Organization and Visualization of Music Archives," In *Proceedings of ACM Multimedia*, pages 570579. ACM 2002.

[3] F. Morchen, A. Ultsch, M. Nocker and C. Stamm: "Databionic Visualization of Music Collections According to Perceptual Distance," In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR-05)*, London, UK. September 2005.

[4] A. Lillie: "MusicBox: Navigating The Space of Your Music," MS thesis, Massachusetts Institute of Technology, MA, USA, 2008.

[5] K. Siedenburg: "An Exploration of Real-Time Visualizations of Musical Timbre," CNMAT, 2009.

[6] P. Toiviainen: "Visualization of Tonal Content with Self-organizing Maps and Self-similarity Matrices," In *Computers in Entertainment (CIE)*, Volume 3, Issue 4, October 2005.

[7] M.Goto: "A Chorus-section Detecting Method for Musical Audio Signals," In *Proceedings of the 2003 IEEE Conference on Acoustics, Speech and Signal Processing*, 2003.

[8] J. Paulus and A. Klapuri: "Music Structure Analysis by Finding Repeated Parts, In *Proc. 1st ACM Audio Music Comput. Multimedia Workshop*, Santa Barbara, CA, Oct. 2006, pp. 5968.

[9] M. Levy, K. Noland and M. Sandler: "A Comparison of Timbral and Harmonic Music Segmentation Algorithms," In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007.

[10] M. Wattenberg: "Arc Diagrams: Visualizing Structure in Strings," In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis02)*, 110, 2002.

[11] J. C. Brown and M. S. Puckette: "An Efficient Algorithm for the Calculation of a Constant Q Transform," *J. Acoust. Soc. Am.,* 92, 2698-2701, 1992.

[12] N. Marwan, M. C. Romano, M. Thiel and J. Kurths: "Recurrence Plots For The Analysis of Complex Systems," In *Physics Reports*, Volume 438, Issues 5-6, January 2007, Pages 237-329.

[13] M. Grachten, W. Goebl, S. Flossmann and G. Widmer: "Phase-plane Representation and Visualization of Gestural Structure in Expressive Timing," In *Journal of New Music Research*, Vol. 38, No. 2, pp. 183-195, 2009.

[14] J. Serra, X. Serra and R. G. Andrzejak: "Cross Recurrence Quantification For Cover Song Identification," In *New Journal of Physics*, Volume 11, 2009.

[15] J. Theiler: "Spurious Dimension From Correlation Algorithms Applied to Limited Time-series Data," *Phys. Rev.*, A 34 (3), 2427-2432, 1986.

[16] R. Spell, R. Brady and F. Dietrich: "BARD: A Visualization Tool For Biological Sequence Analysis," In *Proceedings of InfoVis 2003, IEEE*, pp. 219-225, 2003.

[17] http://www.echonest.com/

[18] http://www.processing.org/

[19] J. R. Mathes: *The Analysis of Musical Form*, Prentice Hall, 2006.

[20] K. Jensen: "Multiple Scale Music Segmentation Using Rhythm, Timbre and Harmony," *Journal on Advances in Signal Processing*, EURASIP, 2007.

A COMPARISON OF PROBABILISTIC MODELS FOR ONLINE PITCH TRACKING

Umut Şimşekli
Boğaziçi University
Department of Computer Engineering
umut.simsekli@boun.edu.tr

A. Taylan Cemgil
Boğaziçi University
Department of Computer Engineering
taylan.cemgil@boun.edu.tr

ABSTRACT

In this study, we propose and compare two probabilistic models for online pitch tracking: Hidden Markov Model and Change Point Model. In our models each note has a certain characteristic spectral shape which we call spectral templates. Hence the system's goal is to find the note whose template is active given the audio data. The main focus on this work is the trade off between latency and accuracy of the pitch tracking system. We present the probabilistic models and the inference schemes in detail. Encouraging results are obtained from the experiments that are done on low-pitched monophonic audio.

1. INTRODUCTION

Pitch tracking is one of the most studied topics in the computer music field since it lies at the center of many applications. It is widely used in phonetics, speech coding, music information retrieval, music transcription, and interactive musical performance systems. It is also used as a pre-processing step in more comprehensive music analysis applications such as chord recognition systems.

Many pitch tracking methods have been presented in the literature. Klapuri proposed an algorithmic approach for multipitch tracking in [1]. Kashino et al. presented applied graphical models for polyphonic pitch tracking [2]. Cemgil presented generative models for both monophonic and polyphonic pitch tracking [3]. Orio et al. and Raphael proposed Hidden Markov Model based pitch tracking methods in [4] and [5] respectively. On the other hand, using nonnegative matrix factorization (NMF) methods become popular at various audio processing applications. Different types of NMF models were proposed and tested on polyphonic music analysis, [6], [7], [8].

In this study, we propose and compare two probabilistic models for online pitch tracking. Our probabilistic models are extensible to polyphonic pitch tracking by using factorial models [9] but we mainly focus on monophonic pitch tracking of low pitched instruments. The main concern of the work is reducing the pitch detection latency without compromising the detection quality. Here the term, latency is defined as the time difference between the true note onset and the time that the pitch tracker has computed its estimate. In our point of view, a pitch tracking method might have latency due to two reasons. The first reason is that the method cannot estimate the note accurately because it has not accumulated enough data yet. The second reason is the computational burden. With the increase of the computational power, the latter can be eliminated by using more powerful computers. Hence, in our work we will focus on decreasing the latency by increasing the accuracy at note onsets rather than trying to reduce the computational complexity. We will test our models on recordings of two low pitched instruments: tuba and bass guitar. This would be challenging since estimating low pitches in shortest time is a difficult problem.

The rest of the paper is organized as follows: in Section 2 and 3 we describe our approach and probabilistic models in detail. We describe our inference scheme in Section 4. The template learning procedure is described in Section 5. In section 6, we present our results on monophonic pitch tracking. Finally Section 7 concludes this paper.

2. APPROACH

In this study, we would like to infer a predefined set of pitch labels from streaming audio data. Our approach to this problem is model based. We will construct two probabilistic generative models that relate a latent event label to the actual audio recording, in this case audio is represented by the magnitude spectrum. We define $x_{\nu,\tau}$ as the magnitude spectrum of the audio data with frequency index ν and time index τ, where $\tau \in \{1, 2, ..., T\}$ and $\nu \in \{1, 2, ..., F\}$.

For each time frame τ, we define an indicator variable r_τ on a discrete state space D_r, which determines the label we are interested in. In our case D_r consists of note labels such as {C4, C#4, D4, D#4, ..., C6}. The indicator variables r_τ are hidden since we do not observe them directly. For online processing, we are interested in the computation of the following posterior quantity, also known as the filtering density [1] :

$$p(r_\tau | x_{1:F,1:\tau}).$$

Similarly, we can also compute the most likely label tra-

[1] Note that we use MATLAB's colon operator syntax in which $(1 : F)$ is equivalent to $[1, 2, 3, ..., F]$.

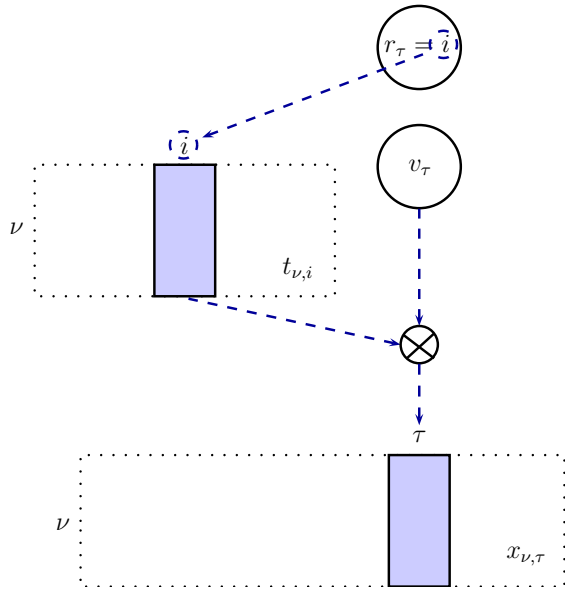


Figure 1. The block diagram of the probabilistic models. The indicator variables, r_τ choose which template to be used. The chosen template is multiplied by the volume parameter v_τ in order to obtain the magnitude spectrum, $x_{\nu,\tau}$.

jectory given all the observations

$$r^*_{1:T} = \underset{r_{1:T}}{\operatorname{argmax}}\, p(r_{1:T}|x_{1:F,1:T}).$$

This latter quantity requires that we accumulate all data and process in a batch fashion. There are also other quantities, called "fixed lag smoothers" that between those two extremes. For example, at time τ we can compute

$$p(r_{\tau-L}|x_{1:F,1:\tau}),$$

where L is a specified lag and it determines the trade off between the accuracy and the latency. By accumulating a few observations from the future, the detection at a specific frame can be eventually improved by introducing a slight latency. Hence we have to fine-tune this parameter in order to have the optimum results.

3. MODELS

In our models, the main idea is that each event has a certain characteristic spectral shape which is rendered by a specific volume. The spectral shapes that we denote as *spectral templates* are denoted by $t_{\nu,i}$. The ν index is again the frequency index and the index i indicates the pitch labels. Here, i takes values between 1 and I, where I is the number of different spectral templates. The volume variables v_τ define the overall amplitude factor, by which the whole template is multiplied. An overall sketch of the model is given in Figure 1.

3.1 The Hidden Markov Model

Hidden Markov Models have been widely studied in various types of applications such as audio processing, natural language processing, and bioinformatics. Like in many computer music applications, HMMs have also been used in pitch tracking applications [4], [5].

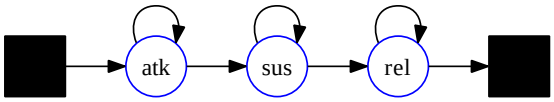


Figure 2. The prior structure of the indicator variable r_τ. Here atk, sus, and rel refers to the attack, sustain, and release parts of a note respectively. The first black square can be either the silence or a note release state. Similarly the second black square can be either a silence or a note attack state.

We define the probabilistic model as follows:

$$\begin{aligned} r_0 &\sim p(r_0) \\ r_\tau|r_{\tau-1} &\sim p(r_\tau|r_{\tau-1}) \\ v_\tau &\sim \mathcal{G}(v_\tau; a_v, b_v) \\ x_{\nu,\tau}|v_\tau, r_\tau &\sim \prod_{i=1}^{I} \mathcal{PO}(x_{\nu,\tau}; t_{\nu,i} v_\tau)^{[r_\tau = i]}, \end{aligned}$$

where the symbols $\mathcal{G}$ and $\mathcal{PO}$ represent the Gamma and the Poisson distributions respectively, where

$$\begin{aligned} \mathcal{G}(v; a, b) &= \exp((a-1)\log v - bv - \log\Gamma(a) + a\log(b)) \\ \mathcal{PO}(x; \lambda) &= \exp(x\log\lambda - \lambda - \log\Gamma(x+1)). \end{aligned}$$

Here we have Markovian prior on the indicator variables, r_τ which means r_τ depends only on $r_{\tau-1}$. We use three states to represent a note: one state for the attack part, one for the sustain part, and one for the release part. We also use a single state in order to represent silence. Figure 2 shows the Markovian structure in more detail.

In some recent work on polyphonic pitch tracking, Poisson observation model was used in the Bayesian non-negative matrix factorization models (NMF) [11]. Since our probabilistic models are similar to NMF models, we choose the Poisson distribution as the observation model. We also choose Gamma prior on v_τ to preserve conjugacy and make use of the scaling property of Gamma distribution.

In this probabilistic model we can integrate out analytically the volume variables, v_τ. It is easy to check that once we do this, provided the templates $t_{\nu,i}$ are already known, the model reduces to a standard Hidden Markov Model (HMM) with a Compound Poisson observation model.

3.2 The Change Point Model

In addition to the HMM, in the change point model (CPM), the volume parameter v_τ has a specific structure which depends on $v_{\tau-1}$ (i.e. staying constant, monotonically increasing or decreasing and etc.). But at certain unknown times, it jumps to a new value independently from $v_{\tau-1}$. We call these times as *"change points"* and the occurrence of a change point is determined by the relationship between r_τ and $r_{\tau-1}$. If $r_{\tau-1}$ jumps to a new value at time τ, in other words if r_τ is not equal to $r_{\tau-1}$, then a change point has occurred at time τ.

The formal definition of the generative model is given below:

$$
\begin{aligned}
v_0 &\sim \mathcal{G}(v_0; a_0, b_0) \\
r_0 &\sim p(r_0) \\
r_\tau | r_{\tau-1} &\sim p(r_\tau | r_{\tau-1}) \\
v_\tau | v_{\tau-1}, r_\tau, r_{\tau-1} &\sim \begin{cases} \delta(v_\tau - \theta(r_\tau) v_{\tau-1}), & r_\tau = r_{\tau-1} \\ \mathcal{G}(v_\tau; a_v, b_v), & r_\tau \neq r_{\tau-1} \end{cases} \\
x_{\nu,\tau} | v_\tau, r_\tau &\sim \prod_{i=1}^{I} \mathcal{PO}(x_{\nu,\tau}; t_{\nu,i} v_\tau)^{[r_\tau = i]}.
\end{aligned}
$$

Here, $\delta(x)$ is the Kronecker delta function which is defined by $\delta(x) = 1$ when $x = 0$, and $\delta(x) = 0$ elsewhere. The $\theta(r_\tau)$ parameter determines the specific structure of the volume variables. Our selection of $\theta(r_\tau)$ is as follows:

$$
\theta(r_\tau) = \begin{cases} \theta_1, & \text{if } r_\tau \text{ is attack,} \\ \theta_2, & \text{if } r_\tau \text{ is sustain,} \\ \theta_3, & \text{if } r_\tau \text{ is release.} \end{cases}
$$

$\theta(r_\tau)$ gives flexibility to the CPM since we can adjust it with respect to the instrument whose sound would be processed (i.e. we can select $\theta(r_\tau) = 1$ for woodwind instruments by assuming the volume of a single note would stay approximately constant). Figure 3 visualizes example templates and synthetic data which are generated from the CPM.

4. INFERENCE

4.1 Inference on the Hidden Markov Model

As we mentioned in Section 3.1, we can integrate out analytically the volume variables, v_τ. Hence, given that the $t_{\nu,i}$ are already known, the model reduces to a standard Hidden Markov Model (HMM) with a Compound Poisson observation model as shown below:

$$
\begin{aligned}
&p(x_{1:F,1:\tau} | r_\tau = i) \\
&\quad = \int dv_\tau \ \exp\Big(\sum_{\nu=1}^{F} \log \mathcal{PO}(x_{\nu,\tau}; v_\tau t_{\nu,i}) \\
&\qquad + \log \mathcal{G}(v_\tau; a_v, b_v)\Big) \\
&\quad = \frac{\Gamma(X_\tau + a_v)}{\Gamma(a_v) \prod_{\nu=1}^{F} \Gamma(x_{\nu,\tau} + 1)} \frac{{b_v}^{a_v} \prod_{\nu=1}^{F} t_{\nu,i}^{x_{\nu,\tau}}}{(T_i + b_v)^{X_\tau + a_v}}.
\end{aligned}
$$

Since we have standard HMM from now on, the inference algorithm can be made to run very fast without any approximation. We can run the well-known forward algorithm in order to compute the filtering density or fixed lag versions with a few backward steps for real-time applications. Also we can estimate the most probable state sequence by running the Viterbi algorithm.

4.2 Inference on the Change Point Model

While making inference on the CPM, our task is finding the posterior probability of the indicator variables, r_τ and volume variables v_τ. If the state space of v_τ, D_v was discrete, then the CPM would reduce to an ordinary HMM on $D_r \times D_v$. However when D_v is continuous, which is our case, an exact forward backward algorithm cannot be implemented in general. This is due to the fact that the prediction density $p(r_\tau, v_\tau | x_{1:F,\tau})$ needs to be computed by integrating over $v_{\tau-1}$ and summing over $r_{\tau-1}$. The summation over $r_{\tau-1}$ renders the prediction density a mixture model where the number of mixture component grow exponentially with τ. In this section we will describe the implementation of exact forward backward algorithm for the CPM and the pruning technique that we use for real-time applications.

The forward backward algorithm is a well known algorithm for computing the marginals of form $p(r_\tau, v_\tau | x_{1:F,\tau})$. We define the following forward messages:

$$
\begin{aligned}
\alpha_{0|0}(r_0, v_0) &= p(r_0, v_0) \\
\alpha_{\tau|\tau-1}(r_\tau, v_\tau) &= p(r_\tau, v_\tau, x_{1:F,1:\tau-1}) \\
\alpha_{\tau|\tau}(r_\tau, v_\tau) &= p(r_\tau, v_\tau, x_{1:F,1:\tau}),
\end{aligned}
$$

where $\tau \in \{1, 2, ..., T\}$. These messages can be computed by the following recursion:

$$
\begin{aligned}
\alpha_{\tau|\tau-1}(r_\tau, v_\tau) &= \sum_{r_{\tau-1}} \int dv_{\tau-1}\, p(r_\tau, v_\tau | r_{\tau-1}, v_{\tau-1}) \\
&\qquad \alpha_{\tau-1|\tau-1}(r_{\tau-1}, v_{\tau-1}) \\
\alpha_{\tau|\tau}(r_\tau, v_\tau) &= p(x_{1:F,\tau} | r_\tau, v_\tau) \alpha_{\tau|\tau-1}(r_\tau, v_\tau).
\end{aligned}
$$

We also define the backward messages and recursions similarly:

$$
\begin{aligned}
\beta_{T|T}(r_T, v_T) &= p(x_{1:F,T} | r_T, v_T) \\
\beta_{\tau|\tau+1}(r_\tau, v_\tau) &= p(x_{1:F,\tau+1:T} | r_\tau, v_\tau) \\
&= \sum_{r_{\tau+1}} \int dv_{\tau+1}\, p(r_{\tau+1}, v_{\tau+1} | r_\tau, v_\tau) \\
&\qquad \beta_{\tau+1|\tau+1}(r_{\tau+1}, v_{\tau+1}) \\
\beta_{\tau|\tau}(r_\tau, v_\tau) &= p(x_{1:F,\tau:T} | r_\tau, v_\tau) \\
&= p(x_{1:F,\tau} | r_\tau, v_\tau) \beta_{\tau|\tau+1}(r_\tau, v_\tau),
\end{aligned}
$$

where $\tau \in \{1, 2, ..., T-1\}$. Moreover, the posterior marginals can simply be obtained by multiplying the forward and backward messages:

$$
\begin{aligned}
p(r_\tau, v_\tau | x_{1:F,1:T}) &\propto p(x_{1:F,1:T}, r_\tau, v_\tau) \\
&= p(x_{1:F,1:\tau-1}, r_\tau, v_\tau) \\
&\qquad p(x_{1:F,\tau:T} | r_\tau, v_\tau, \cancel{x_{1:F,1:\tau-1}}) \\
&= \alpha_{\tau|\tau-1}(r_\tau, v_\tau) \beta_{\tau|\tau}(r_\tau, v_\tau).
\end{aligned}
$$

Due to the fact that r is discrete and v is continuous random variables, in the CPM, we have to store α and β messages as mixtures of Gamma distributions. In order to achieve ease of implementation, we can represent the Gamma mixture

$$
p(v_\tau | r_\tau = i, .) = \sum_{m=1}^{M} \exp(w_m) \mathcal{G}(v_\tau; a_m, b_m),
$$

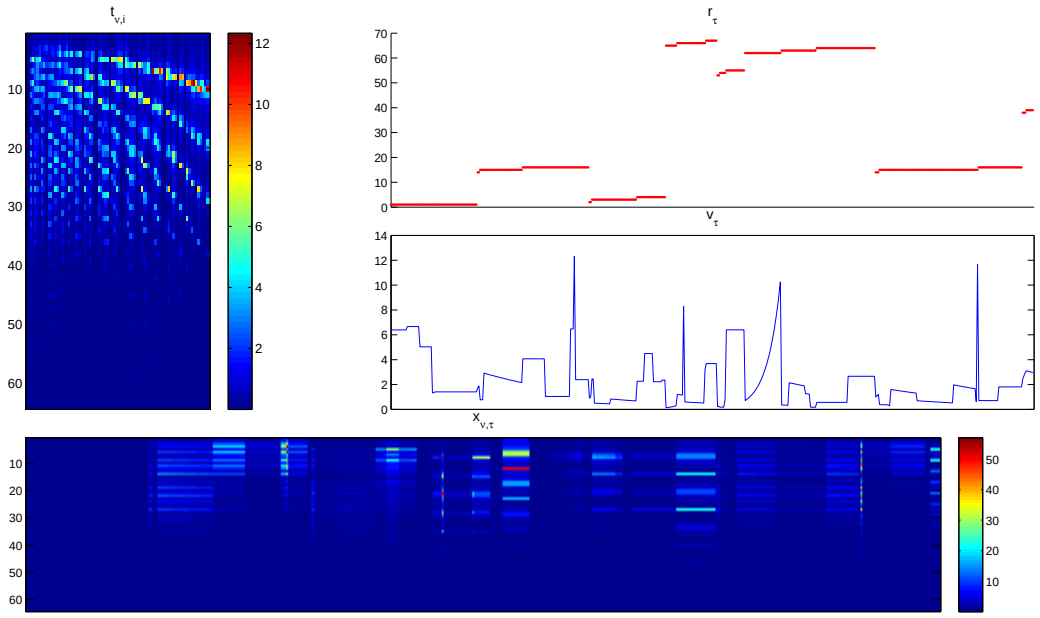


Figure 3. Spectral templates and synthetic data generated from the CPM. It can be observed that the templates implicitly capture the harmonic structure of the signals. The topmost right figure shows a realization of the indicator variables r_τ and the second topmost figure shows a realization of the volume variables v_τ. Here we set $\theta_{1:3} = \{1.10, 0.99, 1.00\}$. With this parametrization, we force the volume variables to increase during the attack parts, slowly damp at the sustain parts and stay constant during the release parts of the notes. The θ parameters should be determined by taking the audio structure into account (i.e. $\theta(r_\tau)$ should be different for higher sustained sounds, percussive sounds, woodwinds, etc.).

as $\{(a_1, b_1, w_1, i), (a_2, b_2, w_2, i), ..., (a_M, b_M, w_M, i)\}$. This will be simply $M \times 4$ array of parameters.

4.2.1 *Forward Pass*

To start the forward recursion, we define

$$\begin{aligned}\alpha_{0|0}(r_0, v_0) &= p(r_0, v_0) \\ &= p(r_0)p(v_0) \\ &= \sum_i^I \exp(l_i)\mathcal{G}(v_0; a_0, b_0),\end{aligned}$$

where, $l_i = \log p(r_0 = i)$. As we mentioned earlier, we represent this message with the array representation of the Gamma mixtures:

$$(a^k_{0|0}, b^k_{0|0}, c^k_{0|0}, d^k_{0|0}) = (a_0, b_0, l_k, k),$$

where k = 1, 2, 3, ..., I denotes the index of the components in the Gamma mixture.

In the forward procedure, we have I Gamma potentials at time $\tau = 0$. Since we are dealing with the CPM, at each time frame, we would have two possibilities: there would be a change point or not. Hence, at $\tau = 1$, we would have I newly initialized Gamma potentials for the possibility of a change point and I Gamma potentials which we copy from the previous time frame, $\tau = 0$, in order to handle the case when a change point does not occur. Similarly, at $\tau = 2$, again we would have I newly initialized Gamma potentials to handle a change point and $2I$ Gamma potentials which we copy from $\tau = 1$. Note that we would have $(\tau + 1)I$ Gamma potentials at time frame τ. Figure 4 visualizes the forward procedure.

Derivation of the prediction step at time τ is as follows:

$$\begin{aligned}&\alpha_{\tau|\tau-1}(v_\tau, r_\tau) \\ &= \sum_{r_{\tau-1}} \int dv_{\tau-1}\, p(v_\tau, r_\tau | v_{\tau-1}, r_{\tau-1}) \\ &\quad \alpha_{\tau-1|\tau-1}(v_{\tau-1}, r_{\tau-1}) \\ &= \sum_{r_{\tau-1}} \int dv_{\tau-1}\, p(v_\tau | r_\tau, v_{\tau-1}, r_{\tau-1}) p(r_\tau | r_{\tau-1}) \\ &\quad \alpha_{\tau-1|\tau-1}(v_{\tau-1}, r_{\tau-1}) \\ &= \sum_{r_{\tau-1}} \int dv_{\tau-1}\, \big([r_\tau \neq r_{\tau-1}]\mathcal{G}(v_\tau; a_v, b_v) \\ &\quad + [r_\tau = r_{\tau-1}]\delta(v_\tau - \theta(r_\tau)v_{\tau-1})\big) p(r_\tau | r_{\tau-1}) \\ &\quad \alpha_{\tau-1|\tau-1}(v_{\tau-1}, r_{\tau-1}).\end{aligned}$$

The first I potentials that handle the change point case become

$$(a^k_{\tau|\tau-1}, b^k_{\tau|\tau-1}, c^k_{\tau|\tau-1}, d^k_{\tau|\tau-1}) = (a_v, b_v, c', k)$$

for k = 1, 2,..., I. Here $a_{ij} = p(r_\tau = i | r_{\tau-1} = j)$ and

$$c' = \log\left(\sum_{\substack{j=1 \\ j\neq k}}^{I} a_{ij} \sum_{m=1}^{\tau I} [d^m_{\tau-1|\tau-1} = j] \exp(c^m_{\tau-1|\tau-1})\right).$$

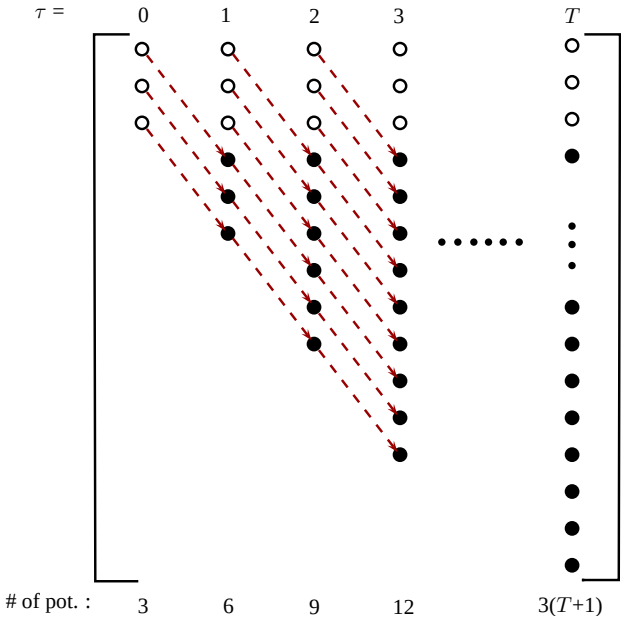


Figure 4. The forward procedure for the CPM where the number of templates, I is 3. The empty circles represent the Gamma potentials that handle the occurrence of a change point and the filled ones handle the other case. The inheritance structure between the time frames are shown with arrows.

We also have τI Gamma potentials which are inherited from the time frame $\tau - 1$:

$$(a^k_{\tau|\tau-1}, b^k_{\tau|\tau-1}, c^k_{\tau|\tau-1}, d^k_{\tau|\tau-1}) = (a^{k-I}_{\tau-1|\tau-1}, \frac{b^{k-I}_{\tau-1|\tau-1}}{\theta(d^{k-I}_{\tau-1|\tau-1})}, c', d^{k-I}_{\tau-1|\tau-1})$$

for k = I+1, I+2,..., $(\tau + 1)I$, where

$$c' = \left(\sum_{i=1}^{I} [d^{k-I}_{\tau-1|\tau-1} = i] \log a_{ii}\right) + c^{k-I}_{\tau-1|\tau-1}$$

Once we compute the predictive distributions, we have to update the Gamma potentials as we acquire the observations:

$$\begin{aligned}
&\alpha_{\tau|\tau}(v_\tau, r_\tau = i) \\
&= p(x_{1:F,1:\tau}, v_\tau, r_\tau = i) \\
&= \alpha_{\tau|\tau-1}(v_\tau, r_\tau = i) p(x_{1:F,\tau}|v_\tau, r_\tau = i) \\
&= \sum_{m=1}^{(\tau+1)I} [d^m_{\tau|\tau-1} = i] e^{(c^m_{\tau|\tau-1})} \mathcal{G}(v_\tau; a^m_{\tau|\tau-1}, b^m_{\tau|\tau-1}) \\
&\quad \prod_{\nu=1}^{F} \prod_{j=1}^{I} \mathcal{PO}(x_{\nu,\tau}; t_{\nu,j} v_\tau)^{[r_\tau = i]}.
\end{aligned}$$

Hence the update equation requires multiplication of Gamma and Poisson potentials. A nice property is that the product is also a Gamma potential, as derived in the Appendix. The updated Gamma potentials are as follows:

$$(a^k_{\tau|\tau}, b^k_{\tau|\tau}, c^k_{\tau|\tau}, d^k_{\tau|\tau}) = (a', b', c', d')$$

for k = 1, 2,..., $(\tau + 1)I$. Here

$$\begin{aligned}
a' &= a^k_{\tau|\tau-1} + \sum_{\nu=1}^{F} x_{\nu,\tau} \\
b' &= b^k_{\tau|\tau-1} + \sum_{i=1}^{I} [d^k_{\tau|\tau-1} = i] \sum_{\nu=1}^{F} t_{\nu,i} \\
c' &= c^k_{\tau|\tau-1} + \sum_{i=1}^{I} [d^k_{\tau|\tau-1} = i] g(a^k_{\tau|\tau-1}, b^k_{\tau|\tau-1}, x, t) \\
d' &= d^k_{\tau|\tau-1}.
\end{aligned}$$

4.2.2 Backward Pass

The backward pass is initialized as follows:

$$\begin{aligned}
\beta_{T|T+1}(v_T, r_T) &= 1 \\
(\hat{a}^k_{T|T+1}, \hat{b}^k_{T|T+1}, \hat{c}^k_{T|T+1}, \hat{d}^k_{T|T+1}) &= (1, 0, 0, k),
\end{aligned}$$

for k = 1, 2,..., I. Here the Gamma potential, $(1, 0, 0, k)$ is the improper Gamma distribution where

$$(a, b, c, k) \times (1, 0, 0, k) = (a, b, c, k),$$

for any a, b, and c.

Similar to the forward pass, we derive the backward recursion as follows:

$$\begin{aligned}
&\beta_{\tau|\tau+1}(v_\tau, r_\tau) \\
&= \sum_{r_{\tau+1}} \int dv_{\tau+1}\, p(v_{\tau+1}, r_{\tau+1}|v_\tau, r_\tau) \\
&\quad \beta_{\tau+1|\tau+1}(v_{\tau+1}, r_{\tau+1}) \\
&= \sum_{r_{\tau+1}} \int dv_{\tau+1}\, p(v_{\tau+1}|r_{\tau+1}, v_\tau, r_\tau) p(r_{\tau+1}|r_\tau) \\
&\quad \beta_{\tau+1|\tau+1}(v_{\tau+1}, r_{\tau+1}) \\
&= \sum_{r_{\tau+1}} \int dv_{\tau+1}\, ([r_{\tau+1} \neq r_\tau] \mathcal{G}(v_{\tau+1}; a_v, b_v) \\
&\quad + [r_{\tau+1} = r_\tau] \delta(v_{\tau+1} - \theta(r_\tau) v_\tau)) \\
&\quad p(r_{\tau+1}|r_\tau) \beta_{\tau+1|\tau+1}(v_{\tau+1}, r_{\tau+1}).
\end{aligned}$$

The backward recursions works very similar to the forward recursions, where we have I potentials at time T. At time $T - 1$, we would have $2I$ Gamma potentials where the first I potentials handle the case of a change point and the remaning I potentials handle the opposite case which is the same case in the forward pass. Note that, in the backward pass we would have τI Gamma potentials at time $(T - \tau)$ as opposed to the forward pass.

4.2.3 The Pruning Procedure

One disadvantage of this model is that the need for the computational power increases as τ increases and exact inference becomes impossible after a couple of steps. In order to eliminate this problem we developed a pruning technique for the CPM as an approximate inference scheme. In the standard pruning algorithms, at time τ, we would sort the Gamma potentials with respect to their mixture coefficients $c^k_{\tau|\tau}$, keep the N best potentials, and discard the

rest of them. However, with this scheme, we may discard the first, immature potentials in the mixture since they have been recently inserted to the mixture.

In this study we propose a different pruning scheme for the CPM. As opposed to the standard pruning methods, we always keep the first N_{keep} Gamma potentials without taking into account their mixture coefficients, where $0 \leq N_{keep} << N$. Then we apply the standard pruning algorithm to the rest of the potentials, i.e. we select the $(N - N_{keep})$ best Gamma potentials.

5. TRAINING AND PARAMETER LEARNING

Since we have constructed our inference algorithms with the assumption of the templates $t_{\nu,i}$ to be known, we have to train the spectral templates at the beginning. In this study we utilized the EM algorithm for this purpose. This algorithm maximizes the log-likelihood iteratively as follows:

$$\begin{aligned}&\text{E-step :}\\&q(v_{1:T}, r_{1:T})^{(n)} = p(v_{1:T}, r_{1:T}|x_{1:F,1:T}, t_{1:F,1:I}^{(n-1)})\\&\text{M-step :}\\&t_{1:F,1:I}^{(n)} = \underset{t_{1:F,1:I}}{\operatorname{argmax}} \left\langle \mathcal{B}^{(n-1)} \right\rangle_{q(v_{1:T}, r_{1:T})^{(n)}}\end{aligned}$$

where

$$\mathcal{B}^{(n)} \quad = \quad p(v_{1:T}, r_{1:T}, x_{1:F,1:T}|t_{1:F,1:I}^{(n)}).$$

The E-step can be computed via the methods which we described in Section 3.1 and 3.2. The M-step for the models is computed as follows:

$$t_{\nu,i}^{(n)} \quad = \quad \frac{\sum_{\tau=1}^{T} \langle [r_\tau = i] \rangle^{(n)} x_{\nu,\tau}}{\sum_{\tau=1}^{T} \langle [r_\tau = i] v_\tau \rangle^{(n)}}.$$

Intuitively, we can interpret this result as the weighted average of the normalized audio spectra with respect to v_τ.

6. EXPERIMENTS AND RESULTS

In order to evaluate the performance of the probabilistic models on pitch tracking, we have conducted several experiments. As mentioned earlier, in this study we focus on the monophonic pitch tracking of low-pitched instruments.

In our experiments we used the electric bass guitar and tuba recordings of the RWC Musical Instrument Sound Database. We first trained the templates offline, and then we tested our models by utilizing the previously learned templates. At the training step, we run the EM algorithm for each note where we use short isolated recordings. On the whole, we use 28 recordings for bass guitar (from E2 to G4) and 27 recordings for tuba (from F2 to G4). The HMM's training phase lasts approximately 30 seconds and the CPM's lasts approximately 2 minutes. At the testing step, we rendered monophonic MIDI files to audio by using the RWC recordings. The total duration of the test files are approximately 4 minutes. At the evaluation step, we compared our estimates with the ground truth which is obtained from the MIDI file. In both our models we used 46 ms. long frames at 44.1 kHz sampling rate.

In our point of view, the main trade-off of these pitch tracking models is between the latency and the accuracy. We can increase the accuracy by accumulating the data, in other words increasing the latency. However after some point the pitch tracking system would be useless due to the high latency. Hence we tried to find the optimum latency and accuracy by adjusting the "lag" parameter of the fixed-lag viterbi path which is defined as:

$$r_\tau^* = \underset{r_\tau}{\operatorname{argmax}}\, p(r_{1:\tau+L}|x_{1:F,1:\tau+L}),$$

where L is the number of audio frames to be accumulated.

As evaluation metrics, we used the recall rate, the precision rate, the computational complexity and the note onset latency. The recall rate, the precision rate and the computational complexity are defined as:

$$\begin{aligned}\text{recall} &= \frac{\text{num. of correct notes}}{\text{num. of true notes}},\\ \text{precision} &= \frac{\text{num. of correct notes}}{\text{num. of transcribed notes}},\\ \text{complexity} &= \frac{\text{running time of the method}}{\text{duration of the test file}},\end{aligned}$$

and we define the note onset latency as the time difference between the pitch tracker's estimate and the ground truth, without considering the label of the estimate. The evaluation results are shown in Figure 5.

We also compared the performance of our models with the YIN algorithm [10]. We used the *aubio* implementation and tuned the onset threshold parameter. The results are shown in Table 1.

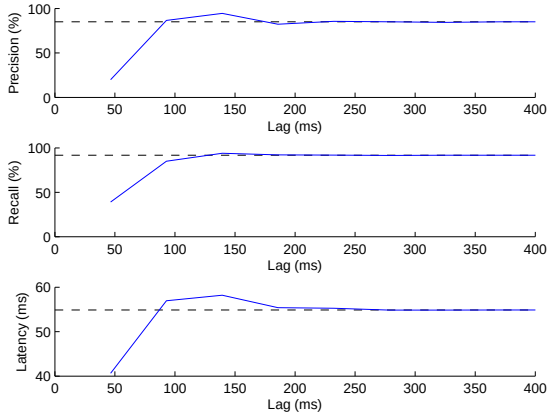
	Rec. (%)	Prec. (%)	Lat (ms)	Comp.
YIN	43.43	9.40	58.74	1.33
HMM	91.72	85.02	54.89	0.02
CPM	97.37	93.59	49.09	0.05

Table 1. The comparison of our models with the YIN algorithm. Here, *Rec*, *Prec*, *Lat* and *Comp* stand for the recall rate, the precision rate, the latency and the computational complexity respectively. The CPM performs better than the others. Moreover, the HMM would also be advantageous due to its cheaper computational needs.

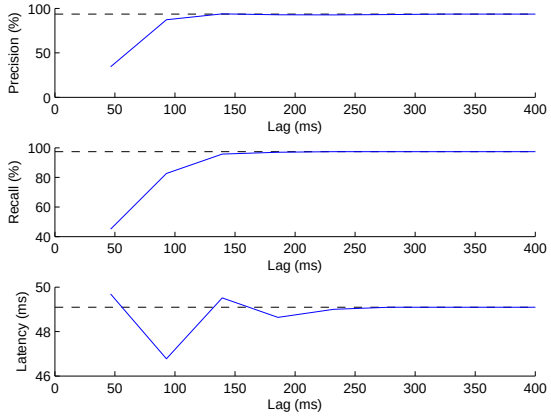
7. DISCUSSIONS AND CONCLUSION

In this study we presented and compared two probabilistic models for online pitch tracking. The main focus was on the trade off between the latency and the accuracy of the proposed pitch detection models.

Apart from the previous works that aimed to develop instrument-independent pitch tracking systems, our approach is based on modeling of a specific musical instrument's spectral structure. Our systems can be optimized for any



(a) HMM performance



(b) CPM performance.

Figure 5. The overall performance of the probabilistic models on low-pitched audio. The dashed lines represent the offline processing results. The total latency of the system is the sum of the lag and the latency at the note onsets.

instrument with a quick training procedure. Besides, this flexible template matching framework can also be used for various types of applications such as acoustic event detection.

Despite testing our probabilistic models on monophonic data, the models are extensible to more complicated scenarios such as polyphony. This kind of extensions require more complex inference schemes, but fortunately there exists powerful state-of-the-art inference methods. Moreover, we can also combine the proposed models with different kinds of probabilistic models for deeper music analysis schemes like joint pitch-tempo tracking.

One limitation of the CPM is that it has the same damping coefficient (θ) for all frequency components in the spectrum. This assumption is limiting since each partial of a note evolves differently over time. As a natural next step of our work is to construct probabilistic models that have frequency dependent damping coefficients.

8. APPENDIX

The update step of a single Gamma potential is derived as follows:

$$
\begin{aligned}
& \log\left(\exp(c)\mathcal{G}(v_\tau;a,b)\prod_{\nu=1}^{F}\mathcal{PO}(x_{\nu,\tau};t_{\nu,i}v_\tau)\right) \\
= \;& c+\log\mathcal{G}(v_\tau;a,b)+\sum_{\nu=1}^{F}\log\mathcal{PO}(x_{\nu,\tau};t_{\nu,i}v_\tau) \\
= \;& c+(a-1)\log v_\tau - bv_\tau - \log\Gamma(a)+a\log(b) \\
& +\sum_{\nu=1}^{F}\left(x_{\nu,\tau}\log t_{\nu,i}v_\tau - t_{\nu,i}v_\tau - \log\Gamma(x_{\nu,\tau}+1)\right) \\
= \;& (a+X_\tau-1)\log v_\tau-(b+T_i)v_\tau+c \\
& -\log\Gamma(a)+a\log(b)+\sum_{\nu=1}^{F}x_{\nu,\tau}\log t_{\nu,i} \\
& -\sum_{\nu=1}^{F}\log\Gamma(x_{\nu,\tau}+1) \\
= \;& (a+X_\tau-1)\log v_\tau-(b+T_i)v_\tau \\
& -\log\Gamma(a+X_\tau)+(a+X_\tau)\log(b+T_i) \\
& +c+g(a,b,x,t) \\
= \;& c'+\log\mathcal{G}(v_\tau;a',b'),
\end{aligned}
$$

where

$$
\begin{aligned}
X_\tau &= \sum_{\nu=1}^{F}x_{\nu,\tau} \\
T_i &= \sum_{\nu=1}^{F}t_{\nu,i} \\
a' &= a+X_\tau \\
b' &= b+T_i \\
c' &= c+g(a,b,x,t)
\end{aligned}
$$

and

$$
\begin{aligned}
g(.) = \;& \log\Gamma(a+X_\tau)-(a+X_\tau)\log(b+T_i) \\
& -\log\Gamma(a)+a\log(b)+\sum_{\nu=1}^{F}x_{\nu,\tau}\log t_{\nu,i} \\
& -\sum_{\nu=1}^{F}\log\Gamma(x_{\nu,\tau}+1)
\end{aligned}
$$

9. ACKNOWLEDGEMENTS

We would like to thank to the three anonymous reviewers for their useful feedback. This work is supported by Bogazici University research fund BAP 09A105P. The work of Umut Şimşekli is supported by the MS scholarship from the Scientific and Technological Research Council of Turkey (TÜBİTAK).

10. REFERENCES

[1] A. Klapuri, "Multipitch analysis of polyphonic music and speech signals using an auditory model," *IEEE*

Transactions on Audio, Speech & Language Processing, vol. 16, no. 2, pp. 255–266, 2008.

[2] K. Kashino, K. Nakadai, T. Kinoshita, and H. Tanaka, "Application of bayesian probability network to music scene analysis," 1998.

[3] A. T. Cemgil, *Bayesian Music Transcription*. PhD thesis, Radboud University of Nijmegen, 2004.

[4] N. Orio and M. S. Sette, "An hmm-based pitch tracker for audio queries," in *ISMIR*, 2003.

[5] C. Raphael, "Automatic transcription of piano music," in *ISMIR*, 2002.

[6] E. Vincent, N. Bertin, and R. Badeau, "Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription," in *ICASSP*, 2008.

[7] N. Bertin, C. Févotte, and R. Badeau, "A tempering approach for itakura-saito non-negative matrix factorization. with application to music transcription," in *ICASSP'09*, 2009.

[8] A. Cont, "Realtime multiple pitch observation using sparse non-negative constraints," in *in International Conference on Music Information Retrieval*, 2006.

[9] A. T. Cemgil, "Sequential inference for Factorial Changepoint Models," in *Nonlinear Statistical Signal Processing Workshop*, (Cambridge, UK), IEEE, 2006.

[10] A. de Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 111, pp. 1917–1930, 2002.

[11] A. T. Cemgil, "Bayesian inference in non-negative matrix factorisation models," *Computational Intelligence and Neuroscience*, no. Article ID 785152, 2009.

DESCRIPTOR-BASED SOUND TEXTURE SAMPLING

Diemo Schwarz
Ircam–CNRS STMS
Paris, France

Norbert Schnell
Ircam–CNRS STMS
Paris, France

ABSTRACT

Existing methods for sound texture synthesis are often concerned with the extension of a given recording, while keeping its overall properties and avoiding artefacts. However, they generally lack controllability of the resulting sound texture. After a review and classification of existing approaches, we propose two methods of statistical modeling of the audio descriptors of texture recordings using histograms and Gaussian mixture models. The models can be interpolated to steer the evolution of the sound texture between different target recordings (e.g. from light to heavy rain). Target descriptor values are stochastically drawn from the statistic models by inverse transform sampling to control corpus-based concatenative synthesis for the final sound generation, that can also be controlled interactively by navigation through the descriptor space. To better cover the target descriptor space, we expand the corpus by automatically generating variants of the source sounds with transformations applied, and storing only the resulting descriptors and the transformation parameters in the corpus.

1. INTRODUCTION

The synthesis of sound textures is an important application for cinema, multimedia creation, games and installations. Sound textures are generally understood as sound that is composed of many micro-events, but whose features are stable on a larger time-scale, such as rain, fire, wind, crowd sounds. We must distinguish this from the notion of *soundscape*, which describes the sum of sounds that compose a scene, each component of which could be a sound texture.

The many existing methods for sound texture synthesis are very often concerned with the extension of a given recording to play arbitrarily long, while keeping its overall properties and avoiding artefacts like looping and audible cut points. However, these methods lack *controllability* of the resulting sound texture. Let's pose an example, that we will use throughout the article: A beginning rainfall, that starts with just a few drops, then thickens, until becoming heavy rain. Even if we have several recordings of the different qualities of rain at our disposal, the existing methods couldn't render the gradual evolution of the rain sound.

To achieve this, we propose a method of statistical modeling of the audio descriptors of texture recordings, that can then be used, varied, or interpolated with other models. Also the steering of the evolution of the generated sound texture is possible, either by giving a target directly in terms of audio descriptors, or deriving these from an existing recording, that couldn't be used directly, e.g. for lack of sound quality or match with the rest of the sound track. Our method is thus strongly based on corpus-based concatenative synthesis (CBCS) [1, 2], which is a new contribution to the field of sound texture synthesis. The use of content-based descriptors is also vastly superior to the often scarce or non-existing meta-data.

CBCS makes it possible to create sound by selecting snippets of a large database of pre-recorded audio (the corpus) by navigating through a space where each snippet is placed according to its sonic character in terms of audio descriptors, which are characteristics extracted from the source sounds such as pitch, loudness, and brilliance, or higher level meta-data attributed to them. This allows one to explore a corpus of sounds interactively or by composing paths in the space, and to create novel timbral evolutions while keeping the fine details of the original sound.

2. RELATED WORK

We will first give an overview of the existing work in sound texture synthesis. As a starting point, Strobl et al. [3] provide an attempt at a definition of sound texture, and an overview of work until 2006. They divide methods into two groups:

Methods from computer graphics Transfer of computer graphics methods for visual texture synthesis applied to sound synthesis [4, 5, 6].

Methods from computer music Synthesis methods from computer music or speech synthesis applied to sound texture synthesis [7, 8, 9, 10, 11].

A newer survey of tools in the larger field of sound design and composition by Misra and Cook [12] follows the same classification as we propose in section 2.1 below. The article makes a point that different classes of sound require different tools (*"A full toolbox means the whole world need not look like a nail!"*).

Filatriau and Arfib [13] review texture synthesis algorithms from the point of view of gesture-controlled instruments, which makes it worthwile to point out the different usage contexts of sound textures:

There is a possible confusion in the literature about the precise signification of the term *sound texture* that is dependent on the intended usage. We can distinguish two frequently occuring usages:

Expressive free synthesis Here, the aim is to interactively generate sound for music composition, performance, or sound art, very often as an expressive digital musical instrument (DMI, e.g. in [13] and [14]). *Sound texture* is then often meant to distinguish the generated sound material from tonal and percussive sound.

The methods employed for expressive texture generation can give rise to naturally sounding textures, as noted by DiScipio [9], but no systematic research on the usable parameter space has been done, and it is up to the user (or player) to constrain herself to the natural part.

Natural texture resynthesis tries to synthesise textures as part of a larger soundscape. Often, a certain degree of realism is striven for (like in photorealistic texture image rendering), but for most applications, either symbolic or impressionistic *credible texture synthesis* is actually sufficient, in that the textures convey the desired ambience or information, e.g. in simulations for urbanistic planning.

2.1 Classification of Synthesis Methods

It seems most appropriate to divide the different approaches to sound texture generation by the synthesis methods (and analysis methods, if applicable) they employ.

Subtractive and additive synthesis, like noise filtering [10, 11, 15] and additive sinusoidal synthesis [16] are the "classic" synthesis methods for sound textures, often based on specific modeling of the source sounds. [1]

Physical modeling can be applied to sound texture synthesis, with the drawback that a model must be specifically developed for each class of sounds to synthesise (e.g. friction, rolling, machine noise) [5, 17], the latter adding an extraction of the impact impulse sound and a perceptual evaluation of the realism of synthesised rolling sounds.

Granular synthesis uses snippets of an original recording, and possibly a statistical model of the (re)composition of the grains [4, 6, 7, 8, 18, 19].

Corpus-based concatenative synthesis can be seen as a content-based extension of granular synthesis [1, 2]. It is a new approach for sound texture synthesis [20, 21, 22]. Notably, Picard [23] uses grain selection driven by a physics engine.

Non-standard synthesis methods, such as fractal synthesis or chaotic maps, are used most often for expressive texture synthesis [9, 13, 14].

There are first attempts to model the higher-level behaviour of whole soundscapes [24], and by using graphs [25, 26].

[1] One venerable attempt is *Practical Synthetic Sound Design* by Andy Farnell at http://obiwannabe.co.uk/tutorials/html/tutorials_main.html.

2.2 Analysis Methods for Sound Textures

Methods that analyse the properties of sound textures are rare, some analyse statistical properties [4, 18, 27, 28], some segment [29] and characterise the source sounds by wavelets [7], and some use adaptive LPC segmentation [30]. Only corpus-based concatenative synthesis methods try to characterise the sonic contents of the source sounds by audio descriptors [1, 2, 20, 21, 31].

3. DESCRIPTOR-BASED SOUND TEXTURE SAMPLING

In order to reproduce a given target sound texture, either with its own sound or by other recordings, we model it by accumulating statistics of its audio descriptor distribution over fixed segments (sizes between 2/3 and 1 second are appropriate, depending on the source sounds).

The descriptors are calculated within the CataRT system [21] by a modular analysis framework [32]. The used descriptors are: fundamental frequency, periodicity, loudness, and a number of spectral descriptors: spectral centroid, sharpness, flatness, high- and mid-frequency energy, high-frequency content, first-order autocorrelation coefficient (expressing spectral tilt), and energy. Details on the descriptors used can be found in [33] and [34]. For each segment, the mean value and standard deviation of each time-varying descriptor is stored in the corpus, although for our example of short segments of static rain sound the standard deviation is not informative.

We evaluated two different methods of statistical modeling: histograms (section 3.1) and Gaussian mixture models (section 3.2).

3.1 Histograms

In the histogram method, the individual distributions of the per-segment descriptor values for an input texture are estimated using histograms.

Figure 1 shows the histograms for three classes of rain for 6 descriptors. The corpus is comprised of 2666 units of length 666 ms in 19 sound files of total length of 29.5 minutes from the *SoundIdeas* database, with 701 units for light rain, 981 for medium rain, and 984 for heavy rain. For this corpus, the descriptors are more or less mutually independent, which means that the conceptually simple histogram method gives acceptable results.

For the control of resynthesis, we use the method known as *inverse transform sampling*, where these histograms are interpreted as probability density functions (PDF), from which we calculate the cumulative sum to obtain the CDF (cumulative density function). We then draw random bin indices accordingly by accessing the CDF by a uniformly distributed random value, and draw a uniformly distributed random descriptor value within the bin in order to generate a stream of target descriptor values that obeys the same distribution as the target, in the limits of the discretisation of the histogram.

The resulting distributions can be easily interpolated to generate a smooth evolution from one texture to the next.

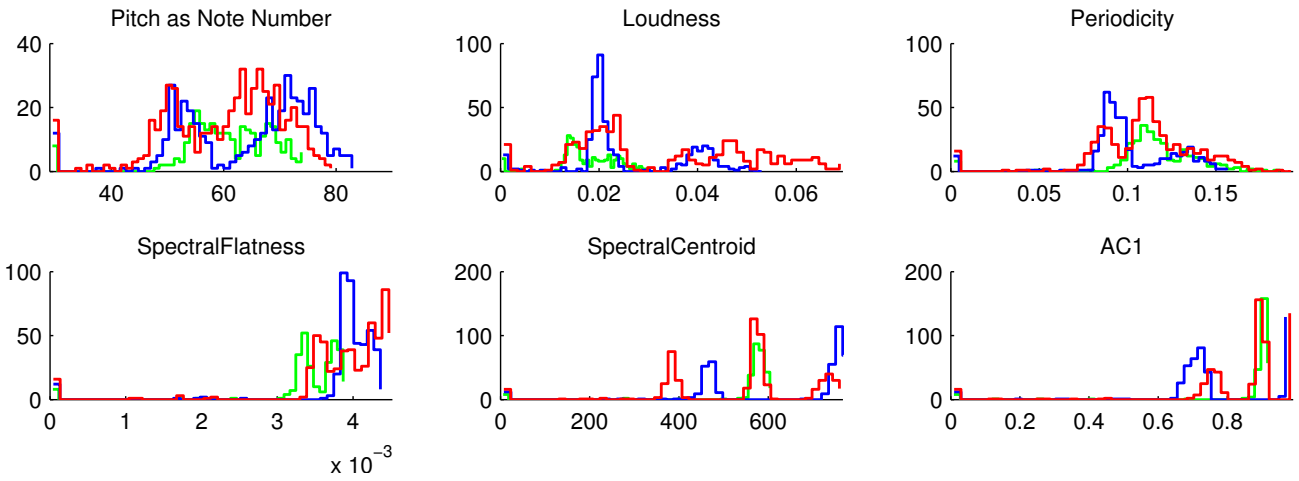


Figure 1. Histograms of spectral centroid, loudness, spectral flatness for the three classes of light (green/clear grey), medium (blue/dark grey), heavy (red/medium grey) rain over a corpus of 2666 segments.

These target descriptors then serve to control a CBCS engine with a corpus of source sounds, as explained in section 3.3.

3.2 Gaussian Mixture Models

In order to capture possible dependencies between the distributions of descriptor values, in this method, we model them by *Gaussian mixture models* (GMMs).

Figure 2 shows the probability density of a two-element mixture for our test corpus, and the interdependencies between two descriptors.

GMMs can be estimated efficiently by Expectation–Maximization. The EM algorithm finds the parameters of a mixture of m multivariate d-dimensional normal distributions:

$$P_j(x|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{\frac{d}{2}} \det(\Sigma_j)^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)} \tag{1}$$

where μ_j are the centres, and Σ_j the covariance matrix. Each mixture component is chosen with a *prior* probability of p_j.

For the control of resynthesis, we first choose the component j of the GMM according to the prior probabilities p_j, and then draw values from the component j by taking advantage of the affine transformation property of normal distributions as

$$P_j = \mu_j + A_j \operatorname{erf}(Z) \tag{2}$$

with Z a uniformly distributed vector, erf the *error function*, i.e. the CDF of a Gaussian, and A_j being the lower triangular matrix from the Cholesky decomposition of Σ_j, i.e. $\Sigma_j = A_j^T A_j$.

GMM parameters can also be interpolated, however, the building of the CDFs for resynthesis is computationally more expensive because of the Cholesky decomposition that needs to be recomputed each time the interpolation changes. Also, care has to be taken to match the m GMM components for the interpolation of μ_j and Σ_j. We chose a greedy matching strategy by closeness of the centres.

3.3 Corpus-Based Concatenative Synthesis

The resynthesis of textures is driven by a vector x of target values for the d used audio descriptors, drawn from the above distributions. Sounds that fulfill these target values are selected from a corpus of source sounds by corpus-based concatenative synthesis, as explained in the following.

The selection of the unit that best matches a given target is performed by evaluating a weighted Euclidean distance C^t that expresses the match between the target x and a database unit u_n with

$$C^t(u_n, x) = \sum_{i=1}^{d} w_i^t \, C_i^t(u_n, x) \tag{3}$$

based on the normalized per-descriptor distances C_i^t for descriptor i between target descriptor value $x(i)$ and database descriptor value $u_i(i)$, normalised by the standard deviation σ_i of this descriptor over the corpus:

$$C_i^t(u_n, x) = \left(\frac{x(i) - u_n(i)}{\sigma_i} \right)^2 \tag{4}$$

Either the unit with minimal distance C^t is selected, or one is randomly chosen from the units within a radius r with $C^t < r^2$, or from the set of the k closest units to the target.

The weights w_j were determined interactively for our test corpus, with equal weights for the spectral descriptors, and half weight for pitch and loudness.

Synthesis is performed by possibly transforming the pitch, amplitude, or timbre of the selected units, and then concatenating them with a short overlap, which is sufficient to avoid artefacts for our texture sounds. One additional transformation is the augmentation of the texture density by triggering at a faster rate than given by the units' length, thus layering several units.

Our synthesis engine (see section 4) works in real time, which allows interactive control of the resulting textures. Therefore, and also because we do not model the transitions between units, the unit selection does not need to use sequence-based matching with the Viterbi algorithm [33].

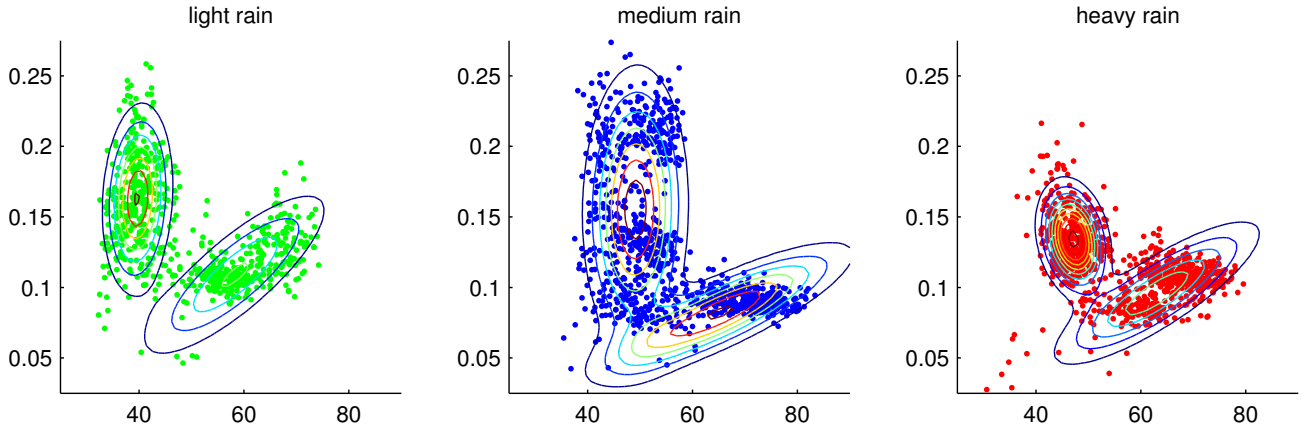


Figure 2. Probability density contours projected on the NoteNumber/Periodicity plane of a Gaussian mixture model of three classes of rain.

3.4 Corpus Expansion

One remaining problem that has not yet been addressed is the possibility that the corpus might not cover the whole range of interpolated and stochastically generated target descriptors. With interactive navigation, we can avoid this shortcoming by judicious tweaking of the playback parameters such as pitch, gain, and filters. In the retargetting case, however, it is hard to derive the necessary transformations from the target values.

This problem could be solved by applying *Feature Modulation Synthesis* (FMS), with the existing research just at its beginning [35]. FMS is concerned with finding the precise sound transformation and its parameters to apply to a given sound, in order to change its descriptor values to match given target descriptors. The difficulty is here that a transformation usually modifies several descriptors at once, e.g. pitch shifting by resampling changes the pitch and the spectral centroid. Recent approaches [36] therefore try to find transformation algorithms that only change one descriptor at a time.

We can get around this problem using a data-driven corpus-based approach, by automatically generating variants of each unit with a certain number and amount of transformations applied, analysing their sound descriptors, and storing only the descriptors and the transformation parameters. The resulting sounds can be easily regenerated on playback.

We generate 5 steps of transpositions by resampling 1 half-tone around the original pitch, and 3 cutoff settings of gentle low-pass and high-pass filters in order to enlarge the timbral variety of the source corpus. The effects of this expansion can be seen in figure 3: a much larger part of the descriptor space between and around the original units is covered by the corpus enlarged 45-fold.

Note that the augmentation of the corpus size does not penalise the runtime of the unit selection much, since we use an efficient kD-tree search algorithm [37] where each doubling of the corpus only adds one more search step on average.

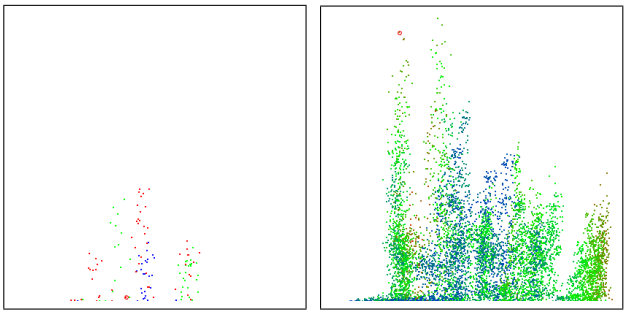


Figure 3. Scatter plot of a texture corpus before (left) and after expansion (right). The x/y/colour axes are spectral centroid, loudness, periodicity.

4. APPLICATIONS AND RESULTS

Our prototype texture synthesiser is implemented in the CataRT system [2] [21] for Max/MSP with the extension libraries FTM&Co [3] [38] making it possible to navigate through a two- or more-dimensional projection of the descriptor space of a sound corpus in real-time, effectively extending granular synthesis by content-based direct access to specific sound characteristics.

The statistical modeling, interpolation, and generation of probability distributions is conveniently handled by the modules mnm.hist, mnm.gmmem, ftm.inter, mnm.pdf from the MnM library [39] included in FTM&Co.

Figure 4 shows an example result using the density parameter, starting from 1 to 10-fold density, resulting in a convincing, albeit quick progression from light rain to a heavy shower. This effect is visible in the gradual whitening of the spectrum. This and other sound examples can be heard on http://demos.concatenative.net.

A creative application of the principle we presented is given in [31], where a musical score for an ensemble was generated from an analysis of sound textures like melting snow or glaciers.

[2] http://imtr.ircam.fr/index.php/CataRT
[3] http://ftm.ircam.fr

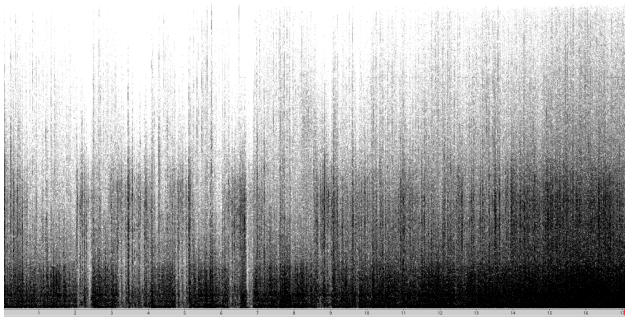


Figure 4. Spectrogram of synthetically densifying rain.

5. CONCLUSION AND FUTURE WORK

The sound textures resulting from our descriptor-driven texture synthesis approach using corpus-based concatenative synthesis stay natural whilst being highly controllable. This goes beyond previous approaches that use an existing recording that is extended in time.

We rely on relatively long segments that capture the fine temporal structure of the sounds, and on crossfade and layering to smooth out timbral changes between units. For less overlap, however, abrupt spectral changes can be noticeable, which could be alleviated in two ways in future work: First, we could take into account the timbral transitions in the selection, avoiding too large jumps in the descriptor space. Second, we could apply the grain segmentation approaches described in section 2.2 and work with the unitary micro-events constituting the source textures (for instance, reconstitute rain by grains of water drop length, cut out of the source sounds).

Code is being developed at the moment that adds a third method of statistical modeling by kernel density estimation. The resulting smoothed d-dimensional histogram captures the interdependencies of the descriptors, unlike the separate histogram method in section 3.1, while allowing a more detailed modeling of the descriptor distribution than GMMs in section 3.2.

The brute-force method of corpus expansion (section 3.4) could be easily optimised by applying a greedy strategy that tries to fill only the "holes" in the descriptor space between existing clusters of sounds. Starting from random transformation parameters, if we hit a hole, we'd explore neighbouring parameters until a desired density of the space is reached.

Finally, the biggest restriction to our modeling approach lies in the assumption of stationarity of the source textures. This is appropriate for many interesting textures, but already rain with intermittent thunder sounds wouldn't be modeled correctly. Clearly, clustering and modeling of the transitions between clusters using hidden Markov models (HMMs) or semi-Markov models seems promising here. This would base the graph approach introduced in [25] on actual data, and could also model the larger-scale temporality of sound scapes as a sequence of textures.

6. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their pertinent comments. The work presented here is partially funded by the *Agence Nationale de la Recherche* within the project *Topophonie*, ANR-09-CORD-022.

7. REFERENCES

[1] D. Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, pp. 3–22, Mar. 2006. Special Issue on Audio Mosaicing.

[2] D. Schwarz, "Corpus-based concatenative synthesis," *IEEE Signal Processing Magazine*, vol. 24, pp. 92–104, Mar. 2007. Special Section: Signal Processing for Sound Synthesis.

[3] G. Strobl, G. Eckel, D. Rocchesso, and S. le Grazie, "Sound texture modeling: A survey," in *Proceedings of the Sound and Music Computing Conference*, 2006.

[4] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Synthesis of audio sound textures by learning and resampling of wavelet trees," *IEEE Computer Graphics and Applications*, vol. 22, no. 4, pp. 38–48, 2002.

[5] J. O'Brien, C. Shen, and C. Gatchalian, "Synthesizing sounds from rigid-body simulations," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 175–181, ACM New York, NY, USA, 2002.

[6] J. Parker and B. Behm, "Creating audio textures by example: tiling and stitching," *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 4, pp. iv–317–iv–320 vol.4, May 2004.

[7] R. Hoskinson and D. Pai, "Manipulation and resynthesis with natural grains," in *Proceedings of the International Computer Music Conference (ICMC)*, (Havana, Cuba), pp. 338–341, Sept. 2001.

[8] C. Bascou and L. Pottier, "GMU, A Flexible Granular Synthesis Environment in Max/MSP," in *Proceedings of the Sound and Music Computing Conference*, Citeseer, 2005.

[9] A. Di Scipio, "Synthesis of environmental sound textures by iterated nonlinear functions," in *Digital Audio Effects (DAFx)*, 1999.

[10] M. Athineos and D. Ellis, "Sound texture modelling with linear prediction in both time and frequency domains," *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 5, pp. V–648–51 vol.5, April 2003.

[11] X. Zhu and L. Wyse, "Sound texture modeling and time-frequency LPC," in *Digital Audio Effects (DAFx)*, vol. 4, 2004.

[12] A. Misra and P. Cook, "Toward synthesized environments: A survey of analysis and synthesis methods for sound designers and composers," in *Proc. ICMC*, 2009.

[13] J. Filatriau and D. Arfib, "Instrumental gestures and sonic textures," in *Proceedings of the International Conference on Sound and Music Computing (SMC)*, 2005.

[14] J. Filatriau, D. Arfib, and J. Couturier, "Using visual textures for sonic textures production and control," in *Digital Audio Effects (DAFx)*, 2006.

[15] M. Lagrange, B. Giordano, P. Depalle, and S. McAdams, "Objective quality measurement of the excitation of impact sounds in a source/filter model," *Acoustical Society of America Journal*, vol. 123, p. 3746, 2008.

[16] S. Guidati and Head Acoustics GmbH, "Auralisation and psychoacoustic evaluation of traffic noise scenarios," *Journal of the Acoustical Society of America*, vol. 123, no. 5, p. 3027, 2008.

[17] E. Murphy, M. Lagrange, G. Scavone, P. Depalle, and C. Guastavino, "Perceptual Evaluation of a Real-time Synthesis Technique for Rolling Sounds," in *Conference on Enactive Interfaces*, (Pisa, Italy), 2008.

[18] Z. El-Yaniv, D. Werman, and S. Dubnov, "Granular Synthesis of Sound Textures using Statistical Learning," in *Proc. ICMC*, 1999.

[19] M. Fröjd and A. Horner, "Sound texture synthesis using an overlap-add/granular synthesis approach," *Journal of the Audio Engineering Society*, vol. 57, no. 1/2, pp. 29–37, 2009.

[20] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," in *Digital Audio Effects (DAFx)*, (Montreal, Canada), Sept. 2006.

[21] D. Schwarz, R. Cahen, and S. Britton, "Principles and applications of interactive corpus-based concatenative synthesis," in *JIM*, (GMEA, Albi, France), Mar. 2008.

[22] M. Cardle, "Automated Sound Editing," tech. rep., Computer Laboratory, University of Cambridge, UK, May 2004.

[23] C. Picard, N. Tsingos, and F. Faure, "Retargetting Example Sounds to Interactive Physics-Driven Animations," in *n AES 35th International Conference, Audio in Games.*, (London Royaume-Uni), 2009.

[24] D. Birchfield, N. Mattar, and H. Sundaram, "Design of a generative model for soundscape creation," in *International Computer Music Conference, Barcelona, Spain*, Citeseer, 2005.

[25] A. Valle, V. Lombardo, and M. Schirosa, "A graph-based system for the dynamic generation of soundscapes," in *Proceedings of the 15th International Conference on Auditory Display*, (Copenhagen), pp. 217–224, 18–21 May 2009.

[26] N. Finney, "Autonomous generation of soundscapes using unstructured sound databases," Master's thesis, MTG, IUA–UPF, Barcelona, Spain, 2009.

[27] M. Desainte-Catherine and P. Hanna, "Statistical approach for sound modeling," in *Digital Audio Effects (DAFx)*, Citeseer, 2000.

[28] C. Bascou and L. Pottier, "New sound decomposition method applied to Granular Synthesis," in *Proc. ICMC*, (Barcelona, Spain), 2005.

[29] S. O'Modhrain and G. Essl, "PebbleBox and CrumbleBag: tactile interfaces for granular synthesis," in *New Interfaces for Musical Expression*, (Singapore), 2004.

[30] I. Kauppinen and K. Roth, "An Adaptive Technique for Modeling Audio Signals," in *Digital Audio Effects (DAFx)*, Citeseer, 2001.

[31] A. Einbond, D. Schwarz, and J. Bresson, "Corpus-based transcription as an approach to the compositional control of timbre," in *Proc. ICMC*, (Montreal, QC, Canada), 2009.

[32] D. Schwarz and N. Schnell, "A modular sound descriptor analysis framework for relaxed-real-time applications," in *Proc. ICMC*, (New York, NY), 2010.

[33] D. Schwarz, *Data-Driven Concatenative Sound Synthesis*. Thèse de doctorat, Université Paris 6 – Pierre et Marie Curie, Paris, 2004.

[34] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the Cuidado project," Tech. Rep. version 1.0, Ircam – Centre Pompidou, Paris, France, Apr. 2004.

[35] M. Hoffman and P. Cook, "Feature-based synthesis: mapping acoustic and perceptual features onto synthesis parameters," in *Proc. ICMC*, (Copenhagen, Denmark), 2006.

[36] T. Park, J. Biguenet, Z. Li, C. Richardson, and T. Scharr, "Feature modulation synthesis (FMS)," in *Proc. ICMC*, (Copenhagen, Denmark), 2007.

[37] D. Schwarz, N. Schnell, and S. Gulluni, "Scalability in content-based navigation of sound databases," in *Proc. ICMC*, (Montreal, QC, Canada), 2009.

[38] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Müller, "FTM—Complex Data Structures for Max," in *Proc. ICMC*, (Barcelona), 2005.

[39] F. Bevilacqua, R. Muller, and N. Schnell, "MnM: a Max/MSP mapping toolbox," in *New Interfaces for Musical Expression*, (Vancouver), pp. 85–88, 2005.

Author Index