LNCS 3902

Richard Kronland-Martinet
Thierry Voinier
Sølvi Ystad (Eds.)

Computer Music Modeling and Retrieval

Third International Symposium, CMMR 2005
Pisa, Italy, September 2005
Revised Papers

Springer

Richard Kronland-Martinet   Thierry Voinier
Sølvi Ystad (Eds.)

Computer Music Modeling and Retrieval

Third International Symposium, CMMR 2005
Pisa, Italy, September 26-28, 2005
Revised Papers

Springer

Volume Editors

Richard Kronland-Martinet
Thierry Voinier
Sølvi Ystad
CNRS-Laboratoire de Mécanique et d'Acoustique
31, chemin Joseph Aiguier, 13402 Marseille Cedex, France
E-mail: {kronland, voinier, ystad}@lma.cnrs-mrs.fr

Library of Congress Control Number: 2006925114

CR Subject Classification (1998): H.3, H.4, H.5, H.2, I.2, C.3

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN	0302-9743
ISBN-10	3-540-34027-0 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-34027-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11751069 06/3142 5 4 3 2 1 0

Preface

This volume constitutes the post-proceedings of the 2005 Computer Music Modeling and Retrieval Symposium (CMMR2005). This event took place during September 26–28, 2005 at the Institute of Information Science and Technologies (ISTI), Italian National Research Council (CNR), Pisa, Italy. CMMR is an annual event focusing on various aspects of computer music. CMMR2005 is the third event in this series. The previous event, CMMR2004, was held in Esbjerg, Denmark, while CMMR2003 was held in Montpellier, France. The CMMR 2003 and CMMR 2004 post-symposium proceedings were published by Springer in the *Lecture Notes in Computer Science* series, LNCS 2771 and LNCS 3310, respectively. CMMR2005 was jointly organized by Laboratoire de Mécanique et d'Acoustique (LMA), Centre National de la Recherche Scientifique (CNRS), Marseille, France and ISTI, CNR, Pisa, Italy.

The field of computer music is interdisciplinary by nature and closely related to a number of computer sciences and engineering areas such as information retrieval, programming, human computer interaction, digital libraries, hypermedia, artificial intelligence, acoustics, signal processing, etc. In this year's CMMR we wanted to emphasize the human interaction in music, simply the PLAY, meaning that papers related to sound modeling, real-time interaction, interactive music, perception and cognition were encouraged. The traditional themes related to information retrieval, programming, digital libraries, and artificial intelligence of course also constituted an important part of the conference as they did in the two previous conferences. The large variability of topics led to fruitful discussions gathering specialists from different fields.

As a novelty in CMMR2005, music contributions were made possible. Various nontraditional, real-time interfaces were presented and used during a concert in the CinemaTeatroLUX in Pisa.

We would first of all like to thank Leonello Tarabella and Graziano Bertini for being the Symposium Chairs, and Massimo Magrini and Stefano Giorgetti for helping with local arrangements and technical support. We would further like to thank the Program Committee members for their crucial paper reports, and all the participants, be it scientists or composers, who contributed with papers and music and made the CMMR2005 a varied and inspiring event. Finally, we would like to thank Springer-Heidelberg for accepting the publication of the CMMR2005 post-proceedings in their LNCS series.

December 2005

Richard Kronland-Martinet
Sølvi Ystad
Thierry Voinier

Organization

CMMR 2005 PLAY! was jointly organized by CNR-Pisa, Italy, and LMA-CNRS Marseille, France.

Symposium Chairs

Leonello Tarabella (CNR, Pisa, Italy)
Graziano Bertini (CNR, Pisa, Italy)

Proceedings Chairs

Richard Kronland-Martinet (CNRS-LMA, Marseille, France)
Sølvi Ystad (CNRS-LMA, Marseille, France)
Thierry Voinier (CNRS-LMA, Marseille, France)

Local Arrangements and Technical Support

Graziano Bertini (CNR, Pisa, Italy)
Stefano Giorgetti (CNR, Pisa, Italy)
Massimo Magrini (CNR, Pisa, Italy)
Leonello Tarabella (CNR, Pisa, Italy)

Program Committee

Program Chair
Sølvi Ystad (CNRS-LMA, Marseille, France)

Members
Jens Arnspang (Aalborg University, Esbjerg, Denmark)
Graziano Bertini (CNR-Pisa, Italy)
Mireille Besson (CNRS-INCM, Marseille, France)
Vittorio Cafagna (University of Salerno, Italy)
Antonio Camurri (University of Genoa, Italy)
Giovanni DePoli (CCS University of Padova, Italy)
Barry Eaglestone (University of Sheffield, UK)
Anders Friberg (Royal Institute of Technology (KTH), Sweden)
Philippe Guillemain (CNRS-LMA, Marseille, France)
Cynthia M. Grund (University of Odense, Denmark)
Goffredo Haus (University of Milan, Italy)

Henkjan Honing (University of Amsterdam, The Netherlands)
Kristoffer Jensen (Aalborg University Esbjerg, Denmark)
Uffe Kock-Wiil (Aalborg University Esbjerg, Denmark)
Richard Kronland-Martinet (CNRS-LMA, Marseille, France)
Marc Leman (University of Gent, Belgium)
Stephen McAdams (Faculty of Music, McGill University, Canada)
Julius Orion Smith III (Stanford University, USA)
Xavier Rodet (IRCAM, Paris, France)
Gary Scavone (Faculty of Music, McGill University, Canada)
Johan Sundberg (Royal Institute of Technology (KTH), Sweden)
Leonello Tarabella (CNR, Pisa, Italy)
Thierry Voinier (CNRS-LMA, Marseille, France)
Gerhard Widmer (University of Vienna, Austria)

Music Selection Chair
Leonello Tarabella (CNR, Pisa, Italy)

Table of Contents

Dynamic Simulation of Note Transitions in Reed Instruments: Application to the Clarinet and the Saxophone

Philippe Guillemain and Jonathan Terroir

CNRS - Laboratoire de Mécanique et d'Acoustique,
31, chemin Joseph Aiguier, 13402, Marseille cedex 20, France
{guillem, terroir}@lma.cnrs-mrs.fr
http://www.lma.cnrs-mrs.fr

Abstract. This paper deals with the simulation of transitions between notes in the context of real-time sound synthesis based on physical models of reed instruments. For that purpose, both the physical and the subjective point of view are considered. From a physical time-varying tonehole model a simple transition model is built, the parameters of which are adapted in order to fit with measurements obtained in normal playing situations. The model is able to reproduce the main perceptive effects, both from the listener point of view, which is a frequency glissando, loudness and brightness variations, and from the player point of view, which is a reduced ease of playing during the transition.

1 Introduction

Sound synthesis based on a modeling of the physical behavior of musical instruments is known to be able to reproduce most of the dynamic aspects of the control of the instruments. It is specifically useful for self-oscillating instruments such as woodwinds or bowed strings instruments, for which the musician has under his control several continuous parameters that act on the pitch, loudness and timbre of the instrument. On these instruments, the simulation of transitions between different pitches is a crucial problem, since a continuous sound production during the closing or opening of toneholes occurs very often in the musical play. For reed instruments, the dynamics of the transition itself can be considered as a part of the musical performance, since the player can decide at what speed the holes are opened or closed. Moreover, transitions have noticeable effects from the point of views of both the player and the listener. The player observes a reduced ease of playing, while the listener may perceive a glissando in frequency during a slow closing or opening, together with variations in the intensity and the brightness of the sound.

In this paper, we present a study the aim of which is to propose a very simple but realistic transition model adapted to the real-time synthesis and control of the clarinet using physical models such as those developed by Guillemain et al. [6]. This study extends the results described in [14] thanks to the comparison with natural sounds and the application of the method to the tenor saxophone.

Experimental signals obtained in usual playing situations of a natural instrument are first considered . In order to get rid of possible artifacts due to a human play, we consider note transitions requiring the closing or opening of a single hole. From the analysis of these signals, we show that the shapes of the frequency glissando, the perceived intensity and the brightness are identical in the closing or opening phases and are independent of the speed at which a tonehole is closed or opened, up to a scaling factor in time.

The next section deals with the physical modeling of the dynamic closing of a tonehole. Many studies have been devoted to the study of toneholes (see for example [9], [11] or [3]). Benade [1] showed that the equivalent length of a single hole pipe is directly function of the geometric properties of the hole. Keefe [10] studied the influence of the size of the holes on frequency, amplitude and playing characteristics. Dalmont and al. [2] have shown how the size of the hole can modify the perceived pitch, timbre and energy. The knowledge of how the hole modifies the behavior of the instrument is essential but what happens during the closing is very important too. Nederveen [12] has considered the influence of the key during the closing, so as Ducasse [4], who has implemented a model of progressive closing of the tonehole, and Scavone and Cook [13] who simulate the closing by varying the reflectance filter of the tonehole between its fully open value and a value nearly equal to one (corresponding to the reflection of the pressure wave at the closed end of the hole). Here, we restrain ourselves to a very simple model, made of a perfectly cylindrical bore and a single, small, opened side-branch with time-varying radius. We show that despite its simplicity, this model is sufficient for the reproduction of the loss of ease of playing, induced by a loss of harmonicity of the impedance peaks and a decrease in the amplitude of the first peaks.

In the next sections, we propose a simplified implementation of this model for use with real-time synthesis, based on an interpolation between the difference equations corresponding to two different resonator lengths. The interpolation coefficients are finally adapted in order to fit with the experimental signals.

The validity of the transition model is checked for conical bores by comparing natural and synthesized saxophone sounds. Last section is devoted to conclusions and perspectives of this work.

2 Experimental Results on the Clarinet

2.1 Experimental Protocol

Measurements have been done on a B-flat clarinet (Yamaha YCL250). In order to observe how the perceived sound is changing during the tonehole closing (or opening), the external pressure has been measured. First, the musician has been asked to close (or open) the first tonehole closed by a “perforated key” (corresponding to the transition from G2 to F2) so as he may close it by bringing closer the finger to the hole until the finger fully closes the hole, as he does during a normal performance, or by sliding his finger so as it progressively reduces the surface of the hole. The second closing technique was used since it can be modelled as a dynamic reduction of the tone hole radius. In order to investigate if the behavior of the transition changes with the closing speed or the kind of action (closing or opening) the musician has been asked to perform the closing and the opening for several speeds (from very slow to very fast). The measurements have been done with an omnidirectional KU81 Neumann in a semi-anechoic room, installed one meter away face to the musician, in order to capture what a listener is hearing.

2.2 Objective Parameters

In this section, we consider objective parameters linked to the behavior of the measured signals, namely the frequencies and amplitude variations of the harmonics during the closing of a tonehole, as functions of the speed or the kind of action (closing or opening).

Frequency of the Harmonics. Figure 1 shows the spectrogram of a measured signal for a slow closing from $G2$ to $F2$ fingering.

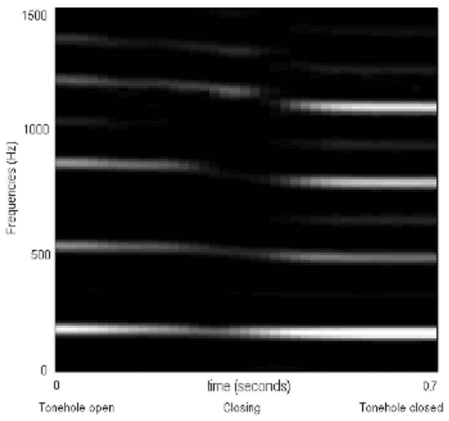


Fig. 1. Spectrogram of a measured external pressure signal obtained for a slow closing from G2 to F2 fingering

Figure 1 clearly shows a frequency glissando. This result is in agreement with the description of the musicians mentioned e.g in [2] or [4]. In order to check if this behavior is the same for each speed or when opening the tonehole, the musician has been asked to close the tonehole with different speeds from very slow to very fast. The figure 2 shows how the frequency of the first harmonic is changing during the closing for four different closing speeds and for one opening (the curve is reversed for an easier comparison). The duration of the closings are $800ms$ and $700ms$ for the very slow closings, $300ms$ for the medium one, $150ms$ for the fast one and $90ms$ for the fast opening. These estimated values correspond to the duration between the two stationary pitches. For each sound, only the closing duration has been considered and an appropriate linear time

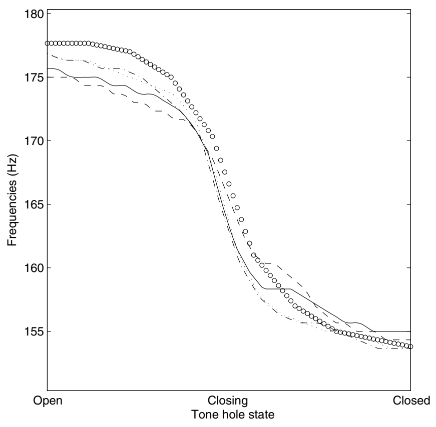


Fig. 2. Variations of the frequency of the first harmonic of measured external pressure signals for different tonehole closing speeds and for a fast opening. Slow closing: solid line and dotted line; intermediate closing speed: dotted line; fast closing: dashed-dotted line; fast opening (reverse curve): circles.

scaling makes it possible to superimpose all the curves in order to allow us to easily compare the different behaviors. One can notice that for each external pressure signal the curves are very similar. Thus, in a synthesis context, this suggests that it is possible to control a transition model only by its duration.

In order to confirm that the frequency of the first harmonic is always varying in a similar way, a second musician has been asked to play some additional sounds with a different instrument. As the first player, the second one recorded slow tonehole closing by sliding his finger or bringing the finger closer to the hole. As it is shown on figure 3, one can notice similar frequencies variations for each musician. Moreover it is important to notice that the two different techniques the musicians have been asked to use are giving similar frequency variations. The frequency variations of the fundamental seems to be a robust parameter always leading to a similar behavior.

Amplitude of the Harmonics. Let us consider now the variations of the amplitudes of the harmonics during the transition. Since the radiation of the instrument depends on the frequency, the directivity and location of the microphone and the position of the musician, the recording conditions (and consequently the listening ones) play a very important role on the amplitude variations of each harmonic that are observed. Moreover, some aspects can have a strong influence on the variations of the level of the harmonics. First, one can mention physical aspects such as localized nonlinearities that may happen for high blowing pressures or direct excitation of the air column at the location of the tonehole induced by the shock of the finger or the key pad for very fast closings. The second aspect is the "human factor". For each recorded sound, there is a variability of the control parameters such as the blowing pressure and the average

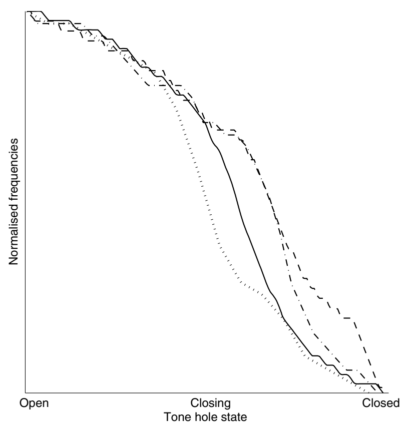


Fig. 3. Variations of the frequency of the first harmonic of measured external pressure signals for different tonehole closing techniques with two different musicians. Solid line: variations obtained by bringing closer the finger to the hole (musician 1); dotted line: variations obtained by sliding the finger on the hole (musician 1); dashed line: variations obtained by bringing closer the finger to the hole (musician 2); dashed-dotted line: variations obtained by sliding the finger on the hole (musician 2).

opening of the reed channel. The musician does not blow or press the reed the same way for each sound he produces. Indeed, after the recordings, we asked the musician to comment the difficulty of the task. The first remark was related to the "unconscious" adjustment of the control parameters in order to prevent the raising of squeaks during slow closings (though the main instruction was to keep the control parameters as constant as possible). The second remark was linked to the way the tonehole closing was done. The instruction was to close it continuously with a constant velocity of the finger, but musicians admitted that because of the sensation on the finger and the risk of squeaks, it was very difficult to keep the velocity constant. For all these reasons, it seems difficult to use the variation of the amplitude of each individual components. Moreover, in a synthesis context, the question is not "does a simplified model acts as a real instrument ?" but rather "what kind of behavior must be in agreement with the perception of natural sounds?". Careful listening to the recorded sounds gives two important and reproducible sensations : the glissando effect previously studied and the sensation that the intensity and the brightness of the sound are first decreasing and then increasing. In order to quantify these variations of the perceived sound intensity, we shall consider the global subjective parameters, rather than the individual levels of the harmonics.

2.3 Subjective Parameters

Loudness

According to Zwicker [18], the subjective parameter directly linked to the perceived intensity of the sound is the loudness. Figure 4 shows how the loudness of the measured signals is varying for the two musicians and several closing techniques. It is worth noticing that the loudness is well defined only for stationary signals. For the purpose of this study, we have considered the non stationary signal during the transition as a succession of stationary signals with a $22ms$ duration. Though the perceptive relevancy of the loudness in such a situation might be questionable, it constitutes a computation tool that allows an objective comparison between signals.

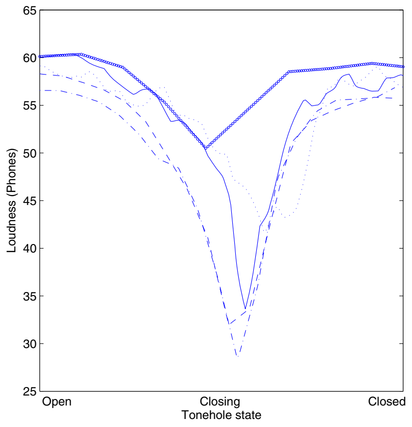


Fig. 4. Variations of the loudness of measured external pressure signals for different tonehole closing speeds and for a fast opening. Slow closing: solid line and dotted line; intermediate closing speed: dashed line; fast closing: dashed-dotted line; fast opening (reverse curve): points.

Similarly to the frequency variations, figure 4 shows that the loudness of the measured signals has a typical shape. One can notice a "valley" which corresponds to a decrease of the perceived intensity. In order to evaluate the influence of the location of the microphone (and consequently of the listener) the sounds have been recorded simultaneously with a directive microphone hanged on the bell of the clarinet. This way, the external pressure is directly measured at the open end of the clarinet. This measure shows that the loudness variations are similar for the two microphones, even though the amplitudes of the harmonics have very different behaviors in the two measures (due to the microphone location and the toneholes radiation).

Brightness

As it has been previously mentioned a careful listening of the sound variation during the tonehole closing shows that the reduction of the loudness comes with a fall of the "brightness" of the sound. Some previous studies have shown that the spectral centroid is one of the parameters directly linked to the notion of "brightness" [5]. The brightness can be expressed as follows (equation 1), with f_i

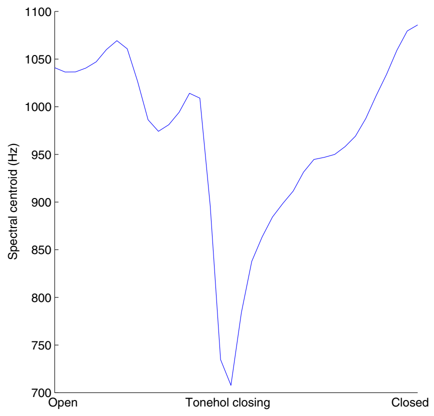


Fig. 5. Variations of the spectral centroid of a measured external pressure signal

the frequency of the i^{th} component, A_i its amplitude and N the number of components taken into account. N corresponds to the last harmonic the amplitude of which is greater than a given threshold.

$$SG = \frac{\sum_{i=1}^{N} f_i A_i}{\sum_{i=1}^{N} A_i} \tag{1}$$

Figure 5 shows the variations of the spectral centroid during a slow tonehole closing performed by a musician. According to him the control parameters have been kept as constant as possible. One can notice that there is a decrease of the spectral centroid during the closing. It means that the brightness of the sound is reduced during the transition. This is in accordance with the perceptive effect we previously mentioned. Moreover this kind of behavior is independent of the technique (sliding / bringing closer), the kind of action (closing / opening) or the speed. Nevertheless it is worth noticing that the values of the spectral centroid depends on the playing conditions (e.g. blowing pressure).

In the next sections, we shall study a physical dynamic tonehole model, simplify its implementation so that it is compatible with real-time synthesis and adapt its parameters in order to reproduce the frequency and loudness variations observed on the natural transition sounds.

3 Simplified Physical Model

3.1 Bore Model

We use a cylindrical pipe with a single hole (see figure 6). It is an oversimplified model of a clarinet body that, according to Benade [1], only considers the role of the first open hole and ignore the others. From a synthesis point of view, such a model seems also sufficient. Indeed, modeling a tonehole lattice with dynamic states different for each tonehole would probably require the introduction of random variables in order to simulate the behavior of human fingers acting simultaneously on multiple keys, and is out of the scope of this paper. In the same way, in order to keep the simplicity of the model, localized losses and non linearities induced by sharp edges or air jets, as well as radiation impedance are ignored here.

Considering electro-acoustical analogy and applying Kirchhoff's law we obtain equations 2 and 3, with p_i the pressures, u_i the flows.

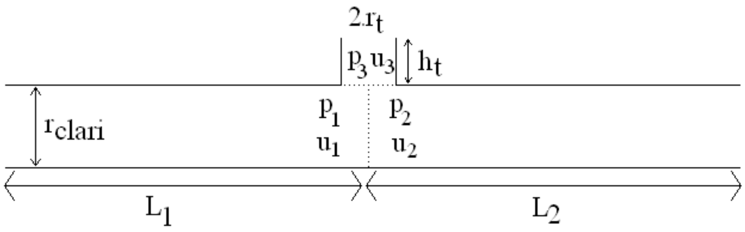


Fig. 6. Cylindrical pipe with a single tonehole. L_1, L_2, r_t, h_t and r_{clari} are the geometrical parameters of the pipe. p_i, u_i are the acoustic pressures and flows.

$$p_1 = p_2 = p_3 \tag{2}$$

$$u_1 = u_2 + u_3 \tag{3}$$

The tonehole of a clarinet is considered as a small cylindrical bore. Its input impedance Z_t, seen from inside the air column and using an ideal open-end termination (the radiation impedance vanishes), is defined as a function of the geometry of the hole (equation 4) with r_t its radius, h_t its length, $S_t = \pi r_t^2$ its surface, $k_t(\omega)$ its wave number (complex-valued in order to include losses), defined classically by assuming that the radius is large in front of the boundary layers thicknesses.

$$Z_t(\omega) = \frac{\rho c}{S_t} j \tan(k_t(\omega) h_t) \tag{4}$$

The dimensionless input impedance, denoted $Z_e(\omega)$, of the whole pipe is a function of the radius of the hole and of the impedances $Z_1(\omega) = jZ_0 \tan(k_1(\omega)L_1)$ and $Z_2(\omega) = jZ_0 \tan(k_2(\omega)L_2)$ corresponding respectively to the two bores of lengths L_1 and L_2 (equation 6), the characteristic impedance of which is $Z_0 = \frac{\rho.c}{S_{clari}}$, where $S_{clari} = \pi r_{clari}^2$ is the surface of the main bore. Again, one assumes that r_{clari} is large in front of the boundary layers thicknesses. The termination impedance $Z_s(\omega)$ of the first part of the bore is made of a parallel combination of the impedances of the tonehole and the second part of the bore. Consequently $Z_s(\omega)$ can be written as equation 5:

$$Z_s(\omega) = \frac{1}{\frac{1}{Z_t(\omega)} + \frac{1}{Z_2(\omega)}} \tag{5}$$

The dimensionless input impedance $Z_e(\omega)$ is obtained classically, using the transmission line theory:

$$Z_e(\omega) = \frac{\frac{Z_s(\omega)}{Z_0} + j\tan(k_1(\omega)L_1)}{1 + j\frac{Z_s(\omega)}{Z_0}\tan(k_1(\omega)L_1)} \tag{6}$$

The closing of the hole is simulated by reducing linearly its radius r_t from a given value corresponding to the totally open hole position to zero. The input impedance $Z_e(\omega)$ is computed for each values of the radius. One can notice that when the tonehole radius equals zero, Z_t is infinite and $Z_s = Z_2$. Consequently, Z_e corresponds to the impedance of a single pipe of length $L_1 + L_2$. Figure 7 shows how the impedance of a single hole pipe ($L_1 = 0.43m$ and $L_2 = 0.05m$, corresponding to a pitch variation from G2 to F2) changes as a function of the radius of the hole. During the closing the equivalent pipe length is varying continuously from L_1 to $L_1 + L_2$. Each horizontal line corresponds to a given value of r_t from $3.5mm$ to zero. The lengths L_1 and L_2 have been chosen in order to correspond to the geometry used for the experiments presented in section 2.

Figure 7 clearly shows that the frequencies of the impedance peaks are sliding.

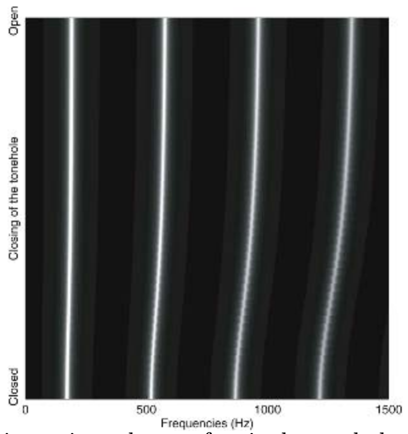


Fig. 7. Variation of the input impedance of a single tonehole pipe as a function of the hole state. The closing is simulated by reducing linearly the hole radius. Opened: the radius r_t is $3.5mm$. Closed: the radius is zero. The radius of the bore r_{clari} is $8mm$.

Figure 8 shows how the inharmonicity of the impedance peaks changes with respect to the tonehole state. This inharmonicity curve is calculated according to equation 7:

$$\tau_{inharm}(n) = \frac{f(n)}{(2n-1)f_0} \tag{7}$$

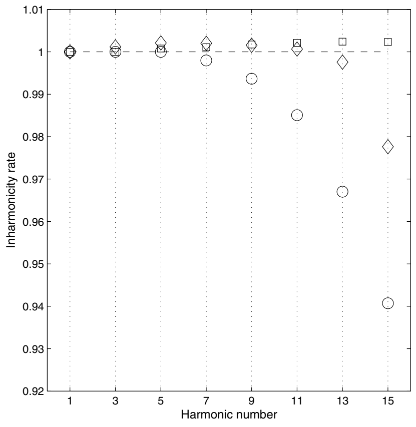


Fig. 8. Inharmonicity of the input impedance peaks obtained for different tonehole states. Totally open tonehole: diamond markers; half-closed tonehole: circle markers; totally closed tonehole: square markers; perfect harmonicity: dashed line.

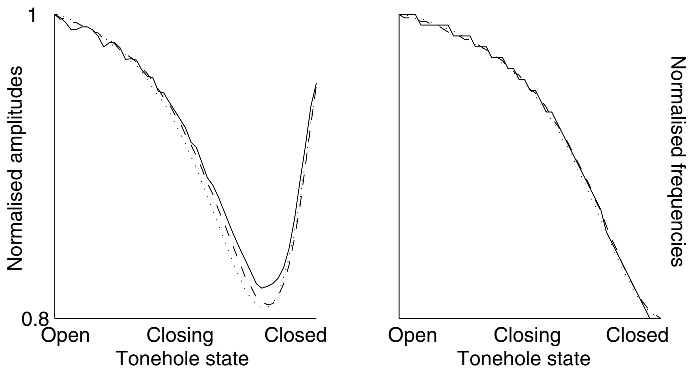


Fig. 9. Normalized amplitude and frequency variations of the first three peaks of the input impedance of a single hole pipe as a function of the tonehole state from totally open to totally closed. The closing is simulated by reducing linearly the hole radius. Peak 1: solid line; peak 2: dotted line; peak 3: dashed-dotted line.

where f_0 is the frequency of the first peak (fundamental) and $f(n)$ is the frequency of the n^{th} peak. Consequently, the value of the inharmonicity is equal to one when the n^{th} harmonic and the fundamental are totally harmonic. The kind of structure we are considering, made of a side branch appended to an uniform tube, is unlikely to have exactly harmonic partials. This is why one observes a more important inharmonicity when the tonehole is on an intermediate state. The inharmonicity observed in the totally opened or totally closed states is caused by the dispersion included in the wave numbers $k_{1-2}(\omega)$.

Figure 9 shows the variations of the amplitude of the sliding impedance peaks. One can observe a decreasing phase followed by an increasing phase.

Inharmonicity combined with lower amplitudes of the peaks has for consequence that the instrument can be more difficult to play, as it as been noticed by Dalmont and al. [2].

Despite its simplicity, this input impedance model is able to reproduce the main phenomena that appear during the tonehole closing : frequency glissando induced by the glissando of the first impedance peak and negative effects on the ease of playing induced by the decrease of the amplitudes of the impedance peaks and the loss of harmonicity. In the next section, we will include this impedance model into a simplified model of the functioning of the whole instrument.

3.2 Full Instrument Model

We consider a classical dimensionless physical model of the functioning of a simplified clarinet-like instrument (see e.g [8]), and its real-time oriented version as proposed in [6]. A description of the method for bores of more complex geometry can be found in [7]. In addition to the impedance described in the previous section (equation 9), the model includes an other linear part that models the reed displacement as a pressure driven mass-spring system (equation 8) and a non-linear part, coupled with the linear parts, that uses the Bernoulli flow model to describe interactions between the flow and the pressure in the mouthpiece (equation 10). The parameter γ is the ratio between pressure inside the mouth of the player and the static beating-reed pressure. ζ, proportional to the square root of the reed opening at rest, characterizes the whole mouthpiece and takes into account the lip position and the section ratio between the mouthpiece and the resonator. x is the reed displacement, u_e the acoustic flow in the mouthpiece, p_e the acoustic pressure in the mouthpiece, and p_{ext} the external pressure. $\omega_r = 2\pi f_r$ and q_r are respectively the circular frequency and the quality factor of the reed.

$$\frac{1}{\omega_r^2}\frac{d^2x(t)}{dt^2} + \frac{q_r}{\omega_r}\frac{dx(t)}{dt} + x(t) = p_e(t) \tag{8}$$

$$P_e(\omega) = Z_e(\omega)U_e(\omega) \tag{9}$$

$$u_e(t) = \frac{1}{2}(1+sign(1-\gamma+x(t)))sign(\gamma-p_e(t))\zeta(1-\gamma+x(t))\sqrt{|\gamma-p_e(t)|} \tag{10}$$

$$p_{ext}(t) = \frac{d}{dt}\left(p_e(t) + u_e(t)\right) \tag{11}$$

The input impedance $Z_e(z)$ (equation 12) is defined in the discrete time domain from equation 6 by the use of the discretization schemes presented in [6]. D_1, D_2, D_{12} are the propagation delays corresponding respectively to the length L_1, L_2 and $L_{tot} = L_1 + L_2$, and $z = \exp(j\omega/f_e)$, where f_e is the sampling frequency. This formulation gives analytically the values of each coefficient of the digital filter as functions of the geometry of the bore and are consequently changing each time the value of r_t is changing.

$$Z_e(z) = \frac{\sum_{k=0}^{k=N_0} bc_k z^{-k} + \sum_{k=1}^{k=N_1} bc_{D1k} z^{-D_1-k}}{1 - \sum_{k=1}^{k=N_0} ac_k z^{-k} - \sum_{k=0}^{k=N_1} ac_{D1k} z^{-D_1-k}} \cdots$$

$$\cdots \frac{+ \sum_{k=0}^{k=N_2} bc_{D2k} z^{-D_2-k} + \sum_{k=0}^{k=N_{12}} bc_{D12k} z^{-D_{12}-k}}{- \sum_{k=0}^{k=N_2} ac_{D2k} z^{-D_2-k} - \sum_{k=0}^{k=N_{12}} ac_{D12k} z^{-D_{12}-k}} \tag{12}$$

Similarly, in the discrete time domain, equation 8 becomes equation 13, where b_{a1}, a_{a1} and a_{a2} are the coefficients of the single mode digital reed model. The digital impedance filter (equation 12) corresponds to equations 17 and 15, where V is a function of the past samples ($p_e(n) = u_e(n) + V$). The non linear coupling (equation 10) is solved analytically by equation 16.

$$x(n) = b_{a1} p_e(n-1) + a_{a1} x(n-1) + a_{a2} x(n-2) \tag{13}$$

$$W = \frac{1}{2}(1 + sign(1 - \gamma + x(n))\zeta(1 - \gamma + x(n)) \tag{14}$$

$$V = f(ac_i, bc_i, u_e, p_e, D_1, D_2, D_{12}) \tag{15}$$

$$u_e(n) = \frac{1}{2} sign(\gamma - V)(-W^2 + W\sqrt{W^2 + 4|\gamma - V|}) \tag{16}$$

$$p_e(n) = u_e(n) + V \tag{17}$$

The sequential use of equations 13, 14, 15, 16, 17 computes at each sample n the values of the variables $x(n)$, $p_e(n)$ and $u_e(n)$, solutions in the digital domain of the discretized problem corresponding to equations 8, 9 and 10. The external pressure $p_{ext}(n)$ corresponding to equation 11 is computed by a difference between the sum of the pressure and the flow at sample n and at sample $n-1$.

Figure 10 shows the spectrogram of the external pressure during the transition. The radius of the tonehole is reduced linearly from $0.35mm$ to $0mm$ ($L_1 = 0.43m$ and $L_2 = 0.05m$). The time of the closing is 0.7 seconds which corresponds to a slow closing. The main effect of the closing, namely the glissando is clearly reproduced. One can notice that, due to the nonlinear coupling with the reed, this glissando is different from that of the first peak of $Z_e(\omega)$. Moreover considering a simple linear reduction of the tonehole radius logically leads to a transition the shape of which is different from the one experimentally observed on figure 1.

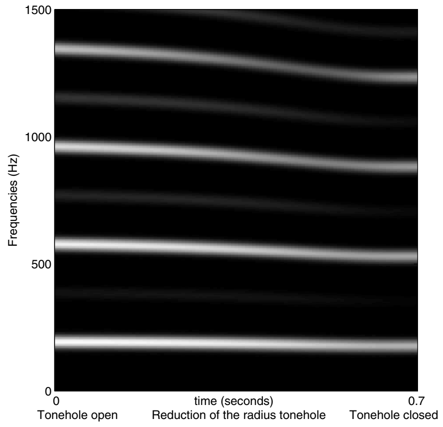


Fig. 10. Spectrogram of the external pressure signal obtained by simulating the closing of the tone hole by reducing linearly its radius from totally open to totally closed. Opened: the radius r_t is $3.5mm$. Closed: the radius is zero. The radius of the bore is $8mm$.

A drawback of this dynamic closing model comes from its numerical complexity. Indeed, with this model, in a real-time implementation, the coefficients of the difference equation linking the output $p_e(n)$ to the input $u_e(n)$, that corresponds to the impedance part of the full model, require to be modified each time the radius of the tonehole is modified. Such a computation cost can become incompatible with a real-time implementation. This is why, in the section 4, we propose a simplified method that approximates the effect of a continuous modification of the input impedance induced by a tonehole closing.

4 Cross-Fade Model

Though the simplified model of a dynamic closing presented in the section 3 gives relevant results, its computational cost might be incompatible with a real-time sound synthesis. Here, we present an alternate method which is compatible with real-time and gives perceptively relevant results. During his study dealing with trumpet modeling, Vergez [15] worked on the problem of the dynamic closing of valves. The problem is similar since one aims at modelling the smooth transition from a length L_1 to a length $L_1 + L_2$ with the introduction of a small side-branch at intermediate states. Vergez chooses to model the closing of a valve by considering simultaneously two resonators during the transition time. He takes into account the change of the effective length by interpolating in the time domain between the two reflection functions corresponding to the two different lengths of the bore. The reflected wave value is an average between the two pressures calculated considering the two reflection functions (the first one for the length L_1, the second one for L_1+L_2). For partially open holes, van Walstijn and Campbell [16] divide the tonehole volume into an "opened part" and a "closed part". The first one behaves as an inertance and the second as a compliance. The impedance of the hole is a function of the impedances of the extreme positions and a parameter defines the ratio between opened and closed states [17].

The discrete-time model we are interested in is based on a digital impedance filter the coefficients of which are explicitly expressed as functions of the geometrical parameters of the instrument. Following similar lines, we interpolate between two filterings corresponding to two different bores, assumed to be perfectly cylindrical. In this case, the difference equation linking $p_e(n)$ to $u_e(n)$ for each bore is given by:

$$p_e(n) = u_e(n) + V \tag{18}$$

and V is given by: $V = -a_1 u_e(n-1) - b_0 u_e(n-D) + a_1 p_e(n-1) - b_0 p_e(n-D)$

To the initial geometry (bore of length L_1) corresponds V_i. To the final geometry (bore of length $L_1 + L_2$) corresponds V_f. During the time of the closing, an interpolation between V_i and V_f is performed. A new value of V, directly function of the values for opened and closed positions is defined as $V = V_i R + V_f R_{inv}$, yielding:

$$p_e(n) = (R + R_{inv}) u_e(n) + V_i R + V_f R_{inv} \tag{19}$$

R and R_{inv} are two time varying functions that define how this cross-fade is performed. They satisfy the relation: $R + R_{inv} = 1$.

From this time domain formulation, we obtain the equivalent formulation in the frequency domain as defined in equation 20, where a_{1i}, b_{0i}, D_i are the coefficients and delays corresponding to the length L_1 and a_{1f}, b_{0f}, D_f those corresponding to the length $L_1 + L_2$.

$$Z(z) = \frac{R + R_{inv} - R(a_{1i}z^{-1} + b_{0i}z^{-D_i}) - R_{inv}(a_{1f}z^{-1} + b_{0f}z^{-D_f})}{1 - R(a_{1i}z^{-1} - b_{0i}z^{-D_i}) - R_{inv}(a_{1f}z^{-1} - b_{0f}z^{-D_f})} \tag{20}$$

Figure 11 shows the equivalent input impedance obtained with this interpolation procedure. Each horizontal line corresponds to a fixed value of R and R_{inv},

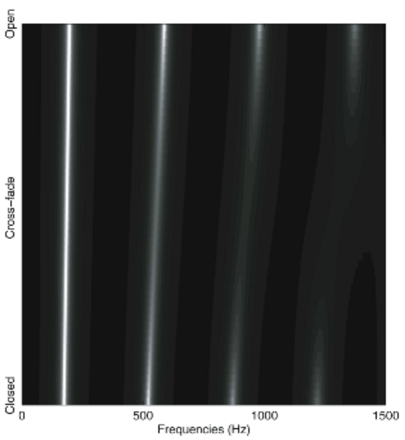


Fig. 11. Variation of the input impedance of a single tonehole pipe obtained by simulating the closing of the hole by a linear cross-fade

and the vertical axis corresponds to a linear decrease of R from 1 to 0. One can notice that the impedance peaks are sliding during the cross-fade.

Figure 12 shows the variations of the heights of the first three impedance peaks. Similarly with the tonehole reduction method, there is a decreasing phase followed by an increasing phase.

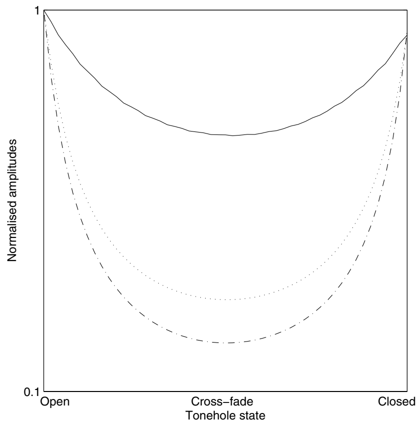


Fig. 12. Variation of the amplitude of the first three impedance peaks of a single tonehole pipe obtained by simulating the closing of the hole with a linear cross-fade. Solid line: harmonic 1; dashed line: harmonic 2; dashed-dotted line: harmonic 3.

The impedance obtained by interpolation and by reduction of the radius of the hole (cf. figure 7) are different. This is not surprising since the interpolation is a very crude approximation. Nevertheless, they share the same important features, since the sliding of the peaks and the behavior of their amplitudes are well reproduced. Moreover, in the synthesis context of this paper, we are interested in the result of the whole model. Let us consider now the external pressure obtained from interpolation of impedances. Figure 13 shows the spectrogram of the external pressure signal obtained by a linear interpolation of impedances. One can notice that the pitch changes non linearly with respect to R and that the level of the harmonics reaches a minimum value around $R = 0.5$. This behavior of the spectrum of the pressure is in good agreement with that obtained with the simplified physical model presented in figure 10. Nevertheless, one can notice that the amplitude and frequency variations obtained by reducing the hole radius and by interpolating linearly are different. This difference makes it necessary to adjust the shape of the function R, by making a correspondence between a given radius of the tonehole r_t and an equivalent value of R [14].

Figure 14 shows how the amplitudes and frequencies of the first harmonics are changing with respect to r_t. From the variations of the frequency of the first harmonic shown in figure 7 and obtained by reducing linearly the radius of the tonehole, one can associate to any given value of r_t yielding a given frequency of the first harmonic, the value of R that leads to the same frequency of the first

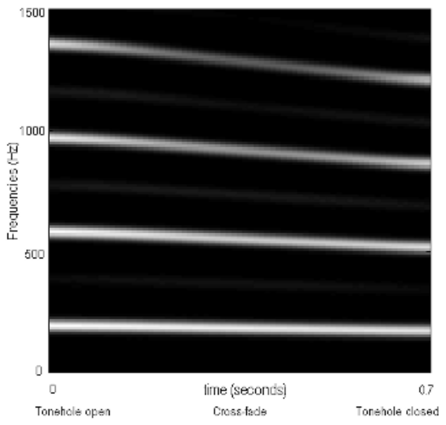


Fig. 13. Spectrogram of the external pressure signal of a single tonehole pipe obtained by simulating the closing of the hole by a linear cross-fade

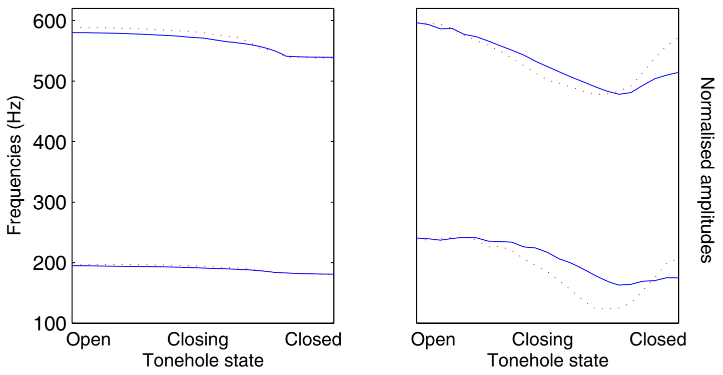


Fig. 14. Amplitude and frequency variations of the first two harmonics of external pressure signals obtained by reducing the tonehole radius (solid line) and by cross-fade with and adapted function R (dotted line). Results are presented as a function of the tonehole state from totally open to totally closed.

harmonic. We focus on this criterion since we are mainly interested in reproducing the perceptive glissando effect of the closing. As shown on figure 14, the use of the adjusted function R_{opti} (and the "inverse" function $R_{opti_{inv}}$) leads to similar frequency variations of the first two harmonics either for the cross-fade method or for the radius reduction method, and in similar variations of the amplitude of the first harmonic. Moreover, this shape is in accordance with the frequency variations of the first peak of the input impedance Z_e of the single hole pipe (see figure 9), which is directly related to the fundamental frequency of the sound.

In normal playing conditions, transitions between notes are obtained by varying continuously r_t over a few milliseconds. It is worth noticing that though this interpolation model constitutes a crude approximation of the physical model presented in section 3, the validity of the physical model itself is questionable in transient situations from both physical and signal processing points of view, since the input impedance (or the reflection function in a wave variable model) and its time-domain equivalent are defined from fixed geometry and stationary hypotheses.

In order to check the validity of the use of this simple interpolation method, we shall compare in the next section the transitions it generates with those obtained from the experimental signals, presented in section 2.

5 Comparison Between Experimental and Simulated Signals

Let us now compare the behavior of the glissando in frequency and the perceptual parameters estimated on experimental signals in section 2 and on simulated signals obtained both by reduction of the radius of the tonehole and by the interpolation method.

Frequency of the Harmonics

We first consider the frequency variations. Figure 15 shows how is changing the fundamental frequency for a measured signal, a simulated signal obtained by reduction of the radius of the tonehole and a simulated signal obtained through the interpolation method. One can notice that the frequency variation of the simulated signal obtained by reduction of the radius of the hole is in good agreement with the frequency variation of the measured signals (as it has been shown before, it is independent of the musician or the closing technique). Since the interpolation model gives frequency variations comparable with that obtained by reducing the radius of the hole, the interpolation method is suitable to reproduce the glissando effect.

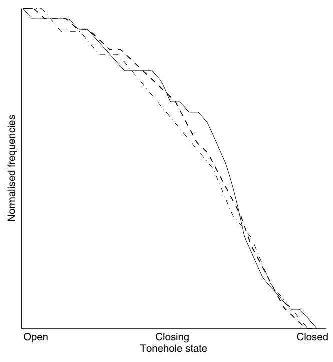


Fig. 15. Variations of the frequency of the first harmonic of a measured external pressure signal (solid line), simulated signal obtained by reducing the tonehole radius (dashed line) and by cross-fade with an adapted function R (dashed-dotted line). Results are presented as a function of the tonehole state from totally open to totally closed.

Loudness

The second important parameter is the loudness. Figure 16 shows the normalized variations of the loudness of two measured signals (closing by bringing closer or by sliding the finger) and two simulated signals (by reduction of the radius of the tonehole or by interpolation between impedances). One can notice that the loudness of the signal obtained by radius reduction method gives a shape which is in agreement with measures. The interpolation method gives a similar result and the "valley" shape is well reproduced. It is obvious that there is a difference with the measurements but the main phenomenon (reduction of the loudness during the closing) is well produced.

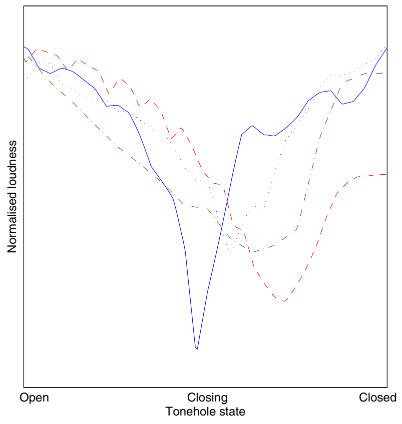


Fig. 16. Variations of the loudness of measured and simulated external pressure signals. Solid line: measured signal obtained by bringing closer the finger to the hole; dotted lines: measured signal obtained by sliding the finger on the hole; dashed-dotted lines: simulated signal obtained by interpolation method; dashed lines: simulated signal obtained by radius reduction method.

Brightness

As it has been previously mentioned the brightness of the sound can be characterized by the spectral centroid. The study of measured signals has shown that the centroid has a typical behavior during the tonehole closing. The measures have also shown that the value of this parameter depends on the control parameters. This makes it difficult to compare the behaviors of measured signals and those of simulated ones. In the context of this study what we are interested in is to obtain a similar behavior of some perceptive parameters for measured and simulated signals. Let us observe how the spectral centroid of simulated signals is evolving for each method. Figure 17 shows the variation of the centroid obtained with the tonehole reduction method (left part) and with the cross-fade method (right part). Similarly to the loudness, the behaviour of the brightness is in accordance with the measurements. Both methods provide centroid variations

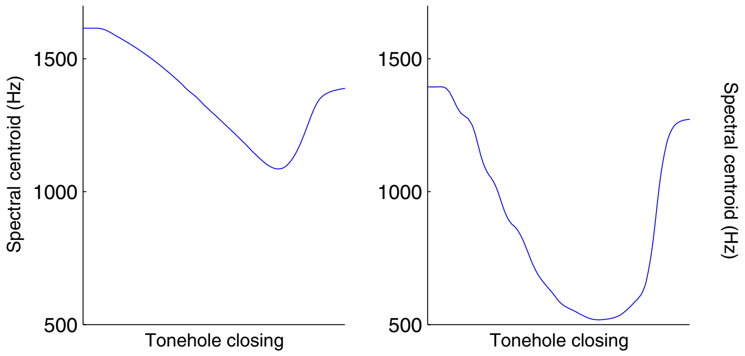


Fig. 17. Variations of the spectral centroid of simulated external pressure signals. Left part: simulated signal obtained by radius reduction method. Right part: simulated signal obtained by interpolation method.

similar to those observed on the measured signals : a fall of the value during the closing. The interpolation method provides a fall of the centroid value which can be considered as accentuated compared to that obtained by the radius reduction method. On the other hand one can notice that with the radius reduction method the final value of the centroid is lower than it could be expected but one has to keep in mind that those simulations are obtained for constant control parameters values.

6 Application to the Tenor Saxophone

Up to now we have considered the clarinet case, the shape of which can be considered as cylindrical. Because of its simplicity and its low computation cost the cross-fade method can be used for more complex geometries. In woodwind instruments, the two main bore shapes are cylindrical and conical. Consequently the next step of this study is to apply the cross-fade method to an instrument the shape of which is conical. Modeling the geometry of a conical bore - side-branch - conical bore is difficult and the application of the radius reduction method would be incompatible with a real-time application. In order to check wether the sound transition is similar in the cases of a cylindrical or a conical bore, measurements have been made on a tenor saxophone Yamaha YTS-475. A musician has been asked to close or open a tonehole (only by bringing closer the key to the hole) for several fingerings, several speeds and several playing conditions (from piano to forte). On the other side, signals have been synthesized using the cross-fade method. It allows to compare the behavior of the real instrument (frequency glissando, loudness, brightness) and the behavior of the simple model for a conical shape. The shape of the function R defining how the cross-fade is performed has been adjusted by making a correspondence with the average frequency variation of the first harmonic of 17 measured signals. Figure 18 displays a saxophone-like bore input impedance computed according to the model described in [7].

Frequency of the Harmonics

The main effect we are interested in is the frequency glissando. Figure 19 shows the frequency variations for measured and simulated signals. As we previously mentioned the shape of the interpolation has been adjusted by making a correspondence with the average frequency variation observed on measured signals. One can notice that the behavior is very similar for the measurement and the simulation.

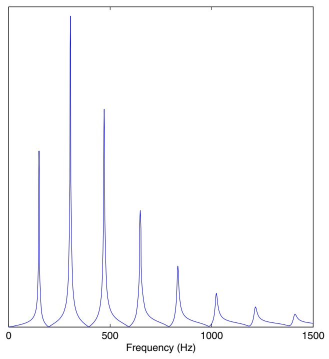


Fig. 18. Saxophone-like bore input impedance

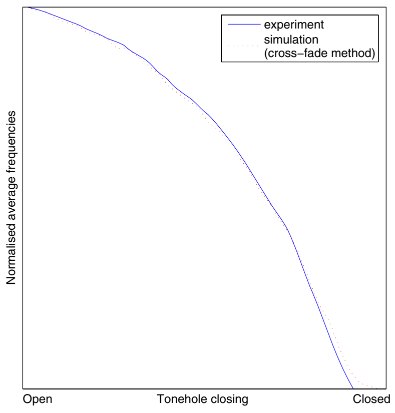


Fig. 19. Frequency variation for a measured tenor saxophone sound (solid line) and a signal obtained by simulating the closing of the tonehole with the cross-fade method (dotted line)

Loudness

Measurements have shown that the loudness behavior can vary a lot with the playing conditions. Playing the tenor saxophone piano or forte induces a very different loudness behavior. For a piano playing, the loudness level is falling during the closing, for a forte playing it does not evolve significantly. Figure 20 shows the variations of the (normalized) loudness of two measured signals (piano (dash-dotted line) and forte (solid line)) and one simulated signal (dotted line) (cross-fade method). One can notice the role of the playing conditions on the loudness variations. Moreover the shape of the simulated curve is typical of what we can observe with the cross-fade model (independently of ζ or γ values). The loudness variations obtained by simulating the tonehole closing with the cross-fade method is not as reliable as we could expect (it's independent of the control parameters) but it does not have an absurd behavior. The simulated loudness variations can be considered as an average behavior. We point out that during the closing the "equivalent" impedance given by equation 20 is not controlled since, for the sake of simplicity, one assumes that $R + R_{inv} = 1$. Obviously during the transition the loudness and the brightness could be controlled by choosing a different relation.

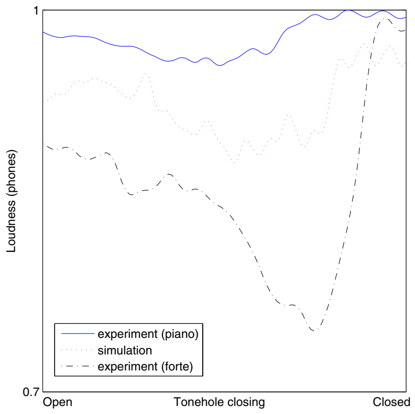


Fig. 20. Variations of the normalized loudness for two measured signals: forte (solid line) and piano (dashed-dotted line) and a signal obtained by simulating the closing of the tonehole with the cross-fade method (dotted line)

Brightness

The third main feature we are interested in is the brightness. As we previously said the study of brightness variations is made by studying the spectral centroid variations. Figure 21 shows the variations of the spectral centroid for a measured signal (left part of the figure) and for a simulated signal (right part). One can notice that the two behaviors are similar : there is an obvious reduction of the spectral centroid frequency during the tonehole closing. Thus the brightness is reduced in the two cases. Since the spectral centroid value depends on the control parameters, one has to focus on the shape of the variations rather than on the value corresponding to the two extreme tonehole states.

One can conclude that the simple cross-fade method is also efficient for a more complex geometry such as a conical one.

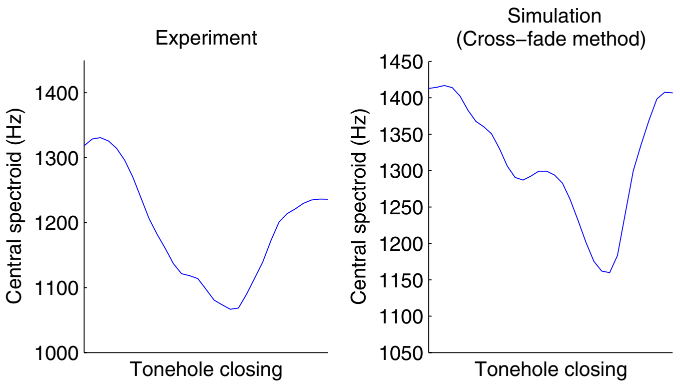


Fig. 21. Variations of the spectral centroid for a measured tenor saxophone signal (left part) and a signal obtained by simulating the closing of the tonehole with the cross-fade method (right part)

7 Conclusions and Perspectives

In this paper, we have presented a very simple interpolation model for the simulation of dynamic transitions between notes on a reed instrument in a real-time synthesis context. The transition laws are adjusted in order to correspond to a single hole resonator model or to measurements. The results of the simulation show a good agreement with the variations measured on natural sounds obtained in normal playing conditions. In particular, it has been observed that, on a real clarinet, the loudness, brightness and pitch variations during a transition requiring the closing of a single tonehole are, up to a time dilation, independent of the transition speed. This allows a simple piloting of the transition model with the velocity of the closing. The validity of the model has been checked for more complex bore geometries (e.g. conical bores) using natural saxophone sounds. The model can be improved through analytic formulations of the frequency and amplitude variations of the first impedance peak during the closing in order to obtain analytic expressions of the interpolation functions. Thanks to experiments with an artificial mouth and impedance measurements, these methods will be improved under more calibrated playing conditions. The use of this transition model for various wind instruments can be found at: http://www.lma.cnrs-mrs.fr/∼guillemain/index.html.

Acknowledgements

The authors would like to thank Richard Kronland-Martinet for playing the natural clarinet sounds, Thierry Voinier for their recording and Jean Kergomard for helpfull discussions on the physical models.

References

1. Benade, A. H.: Fundamentals of musical acoustics.zzzzzz Oxford University Press London (1976)
2. Dalmont, J.P., Ducasse, E., Ollivier, S.: Practical consequences of toneholes non-linear behavior. Proceedings of the ISMA 2001 Perugia Italia (2001)
3. Dubos, V., Kergomard, J., Khettabi, A., Dalmont, J.P., Keefe, D.H. , Nedervenn, C.J.: Theory of sound propagation in a duct with a branched tube using modal decomposition. Acustica - Acta Acustica Vol.85 (1999) 153–169
4. Ducasse, J.: Modélisation et simulation dans le domaine temporel d'instruments à vent à anche simple en situation de jeu : méthodes et modèles. pHd Thesis Université du Maine (2001)
5. Grey, J.M. and Gordon, J.W.: Perceptual effects of spectral modifications on musical timbres. J. Acoust. Soc. Am., Vol.63, (1978) 1493–1500"
6. Guillemain, P., Kergomard, J., Voinier, T.: Real-time synthesis of clarinet-like instruments using digital impedance models. J. Acout. Soc. Am, Vol.118(1) (2005) 483–494
7. Guillemain, P.: A digital synthesis model of double-reed wind instruments. Eurasip Journal on Applied Signal Processing, Special Issue on Model-based Sound Synthesis, Vol. 4(7), (2004) 990–1000
8. Hirschberg, A., Kergomard J., Weinreich, G.: Mechanic of musical instruments. CISM courses and lectures no. 355, Springer-Verlag, Vienne, (1995)
9. Keefe, D.H.: Theory of the single woodwind tonehole. J. Acoust. Soc. Am. Vol.72(3) (1983) 676–687
10. Keefe, D.H.: Acoustic streaming, dimensional analysis of nonlinearities, and tonehole mutual interactions in woodwinds. J. Acoust. Soc. Am. Vol.73 (1983) 1804–1820
11. Nederveen, C.J., Jansen, J.K.M., Van Hassel, R.R.: Corrections for woodwind tonehole calculations. Acta Acustica Vol.84 (1998) 657–699
12. Nederveen, C.J.: Acoustical aspects of woodwind instruments, revised edition. Northern Illinois University Press Illinois (1998)
13. Scavone, G., Cook, P.: Real-time computer modeling of woodwind instruments. Proceedings of the ISMA 1998 Leavenworth WA USA (1998)
14. Terroir, J., Guillemain, P.: A simple dynamic tonehole model for real-time synthesis of clarinet-like instruments. ICMC 2005 Proceedings, Barcelona, Spain (2005)
15. Vergez, C.: Trompette et trompettiste : un système dynamique non linéaire à analyser, modéliser et simuler dans un contexte musical. pHd Thesis Université Paris 6 (2000)
16. van Walstijn, M., Campbell, M.: Discrete-time modeling of woodwind instrument bores using wave variables. J. Acoust. Soc. Am. Vol.113(1) (2003) 575–585
17. van Walstijn, M., Scavone, G.: The wave digital tonehole model. Proceedings of the ICMC 2000 Berlin Germany (2000)
18. Zwicker, E., Feldtkeller, R. Das Ohr als Nachrichtenempfnger. Stuttgart. S. Hirtzel Verlag (1967)

The BRASS Project, from Physical Models to Virtual Musical Instruments: Playability Issues

Christophe Vergez[1,2] and Patrice Tisserand[2]

[1] LMA-CNRS, 31 Ch. Joseph Aiguier, 13402 Marseille Cedex 20, France
[2] IRCAM, 1 Pl. Igor Stravinsky, 75004 Paris, France
vergez@lma.cnrs-mrs.fr
tisserand@ircam.fr
http://www.ircam.fr, http://www.lma.cnrs-mrs.fr

Abstract. The Brass project aims to deliver software virtual musical instruments (trumpet, trombone, tenor saxophone) based on physical modelling. This requires to work on some aspects of the playability of the models so that they can be played in real time through a simple keyboard : better control of the attacks, automatic tuning, humanization.

1 Introduction

Sound synthesis by physical modeling has been developping for more than thirty years. Various models and digitizing techniques are available (see [1] and [2] for a complete review), and commercial applications are proposed since the mid-ninetees[1].

The goal of the Brass project is to propose software virtual musical instruments (based on physical models) playable in real-time and controlled through a simple keyboard. Target instruments included in the project are the trumpet, the trombone, and the tenor saxophone[2]. This project has been done at Ircam[3] and Arturia[4] and is supported by the riam[5] network. This paper is not a review of the whole project, but focuses more specifically on the strategies implemented to improve the playability of the models. Therefore, the physical models considered as well as numerical techniques used for the implementation are not described in this paper.

For a sound synthesis software to achieve the characteristics of a virtual instrument, two key features are needed :

[1] A well known example is the Yamaha VL1 keyboard based on digital waveguides developped at CCRMA, Stanford University.
[2] Which is obviously not however a *brass* instrument.
[3] Ircam people involved sorted by name : A. Almeida, R. Caussé, X. Rodet, N. Schnell, P. Tisserand, C. Vergez (ext. coll. LMA).
[4] Arturia people involved sorted by name : F. Bourgeois, Y. Bonnefoy, N. Bronnec, J. Germond, X. Oudin, F. Paumier, N. Pianfetti, S. Simmermacher.
[5] http://www.riam.org/riam/

– The first one is obviously the sound synthesis technique, which has to be flexible enough to provide natural variations of the sound when input parameters (controlled by the player) are altered. To do this, physical modeling has been chosed. The models are briefly discussed in section 2.
– The second key feature, which is more highlighted in this paper, is the playability of the software application. Indeed, significant effort has to be provided in order to transform a physical model (even if it has the intrinsic ability to produce typical sound effects of a given instrument) into an easily playable virtual instrument. In fact, unlike in a musical acoustics study, where squeaks and slangs are welcome since they highlight the ability of the model to reproduce typical features of the instrument, they have to be avoided here. Indeed, the main goal of the project is to immediately make the keyboardist feel like if he knew how to play well the instrument. Moreover, the player is supposed to have no other MIDI controler than those provided by a standard mid-range synthesizer. Therefore, significant efforts were put on the fine tuning of the model, so that each key pressed on the keyboard makes the model play the desired note (see section 3). This is far from being a straightforward task with a physical model. Finally, in order to overcome limitations generated by the control of a model through a keyboard, an additional layer of (higher level) control has been added between the keyboard output and the model input (see section 4). This can be seen as an attempt to propose gesture models for attack, vibrato, legato ...

2 Physical Modelling

2.1 General Principles

The physical models used in the BRASS project rely on a formulation of the physical functionning principles in term of nonlinear delay differential equations (popularized for sound synthesis of self-sustained musical instruments by [3]). The trumpet and the trombone models have been mainly developped during the Phd thesis of the first author (see [4], [5] for a general description, and [6], [7] for precise aspects of the models) starting from the physical description given (among others) by Elliott and Bowsher ([8]). The saxophone model has been developped during the ongoing Phd of André Almeida on reed instruments (similar modelling principles applied to the oboe can be found in [9]). However, this paper is more specifically devoted to the trumpet and the trombone.

2.2 Modified Model for the Lips

The classical single-mass lips model for the trumpet and the trombone has been slightly modified in order to obtain (and control) more typical brass sounds during the attack transient. Since the computational cost had to be kept as low as possible, the additional complexity of a two-mass model ([10], [11]) or even a single-mass model with two degrees of freedom ([12], [13], [14]) could not be afforded. Moreover, since the steady-state behavior of the model was satisfying, the modified model differs only in the first milliseconds of the sound, during the transient.

The new model proposes to take into account the influence of the tongue at the attack onset. In any brass instrument, the note starts when the tongue stops to stick to the lips and let the path free for the air flow. In spite of poor agreement between brass players on the precise tongue movement, its critical influence on the transient characteristics is well acknowledged. In this study, while the tongue itself has not been modelled, its influence on the lips dynamics is considered: the quick removal of the tongue is supposed to generate a disturbance force on the mass of the form:

$$F = F_l' e^{-\alpha_l' t} \sin \omega_l' t \qquad (1)$$

Equation (1) is applied until time[6] $t^* \triangleq \frac{-1}{\alpha_l'} \ln \frac{10^{-3}}{F_l'}$. The particular expression of F may obviously be thought as the impulse response of a second order damped system. However, equation (1) should be seen as an additional way of controling the attack characteristics (depending on values of F_l', ω_l' and α_l') rather than a refined physical model deriving from a rigorous analysis. However, from the point of view of perception, the flexibility added might be associated with the different types of attack transients a brass player can produce using various tongue techniques (from a soft attack without any use of the tongue, to very pronounced attacks). In figure 1, two attack transients (note G4) are synchronized to highlight the influence of F (equation (1)) both on the length of the attack and on

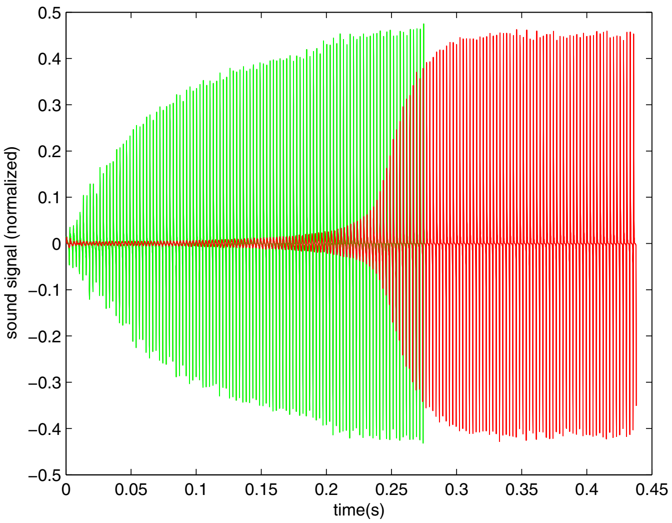


Fig. 1. Comparison of two transients (note G4): green one corresponds to $F_l' = 0$ in (1), red one corresponds to $F_l' = 3$, $\omega_l' = \omega_l$, $\alpha_l' = 350$, $t^* = 22$ms

[6] t^* corresponds to the time where the amplitude of the force F as descreased by a factor 1000 until $t = 0$.

the shape of the enveloppe. Note that the additional force (equation (1)) is only taken into account until $t^* = 22ms$ in figure 1.

2.3 Control Parameters of the Model

Finally, the input parameters for the trumpet and the trombone models are given in table 1. They are divided into two groups : the tuning parameters and the control parameters. Values of the parameters inside the first group are automatically determined (see section 3). On the contrary, parameters inside the second group can be mapped by the player to various controllers.

Table 1. Input parameters of the model for the trumpet/trombone

Tuning parameters	
Name	Description
ω_l	Resonance frequency of the lips
α_l	Damping of the lips
l_b	Length of the bore (i.e. valves position for the trumpet, and length of the slide for the trombone)
$\left\{A^i, \Omega^i\right\}_{i=1,2}$	Amplitude and Frequency of the additional modes introduced to compensate for the truncation of the reflection function (see [4] for details)
Control parameters	
Name	Descrition
p_m	Mouth (or blowing) pressure
λ	Amount of losses within the bore of the instrument
ϵ	Amount of randomness injected in the air flow
$\Delta\omega_l$, $\Delta\alpha_l$	Variations around ω_l and α_l

3 Automatic Tuning of the Physical Models

3.1 Problem Statement

Physical models discussed in section 2 have proved to be well adapted for sound synthesis. However, the trumpet and the trombone models are particularly difficult to tune. Indeed, as it is well known since [15], the lips modal characteristics (ω_l, α_l) influences significantly the frequency of the auto-oscillation.

Therefore, in order to avoid tedious adjustments (by hand) of the tuning parameters (listed[7] in the upper part of table 1), the tuning is considered as an optimization procedure : we are looking for the parameter values which make the model produce the sound whose fundamental frequency f_0 matches at best a target frequency $\hat{f}_0$. This comes to minimize a cost function (see sections 3.2 and 3.3). A general flowchart of the tuning method is presented in figure 2.

[7] For certain notes, the length of the bore is determined by the tuning process in order to allow fine adjustments of the playing frequency, just as a trumpet player does with the tuning slide.

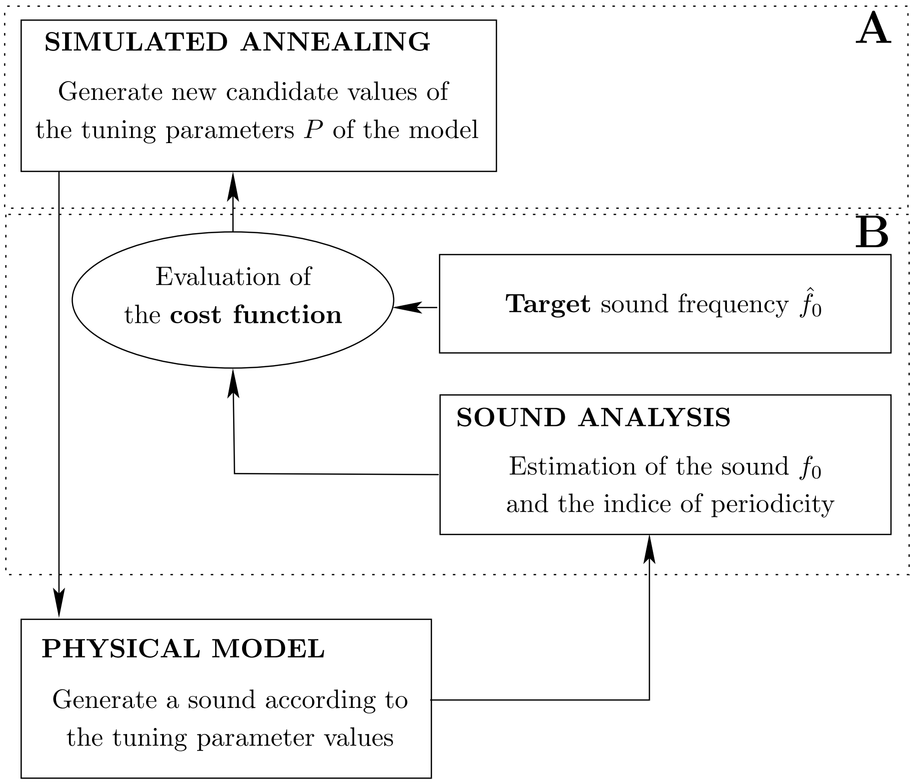

A
SIMULATED ANNEALING
Generate new candidate values of the tuning parameters P of the model
B
Evaluation of the **cost function**
Target sound frequency $\hat{f}_0$
SOUND ANALYSIS
Estimation of the sound f_0 and the indice of periodicity
PHYSICAL MODEL
Generate a sound according to the tuning parameter values

Fig. 2. Basic principle of the automatic tuning procedure by numerical optimization. The dotted squares marked **A** and **B** correspond to section 3.3 and 3.2 respectively.

3.2 Cost Function

The cost function $\mathcal{C}$ is a measure of the discrepancy between the target response $\hat{f}_0$ and the response f_0 of the model to candidate parameter values P proposed by the optimisation process (detailed later in section 3.3). Therefore, the cost function $\mathcal{C}$ may have many different expressions. The most intuitive expression of $\mathcal{C}$ depending only on the square or the absolute value of $(f_0 - \hat{f}_0)$ could not be retained. As a matter of fact, among all possible parameter values, only few of them make the model produce periodic sounds. Therefore, the cost function has to be penalized by a periodicity criteria. This is done by using a monophonic version of the fundamental frequency extractor recently developped at Ircam ([16]) which also calculates a confidence rating of the proposed f_0 (between 0 and 1), further used in this paper as a periodicity descriptor d_p (the more d_p is close to 0, the more we penalize the cost function). Finally the cost function defined by (2) proved to be efficient to match the target $\hat{f}_0$, while avoiding quasi-periodic and non harmonic sounds (see figure 3 for a graphical representation).

$$\mathcal{C}\left(\hat{f}_0, f_0\right) = e^3\left(1 - e^{3(d_p-1)}\right) + |\hat{f}_0 - f_0| \qquad (2)$$

A probabilistic technique has been chosen to minimize the above cost function (see section 3.3), because it appeared to have many local minima. As a matter of fact, as far as its minimization is concerned, variables of the cost function are the tuning parameters (and not f_0 and d_p as shown in figure 3).

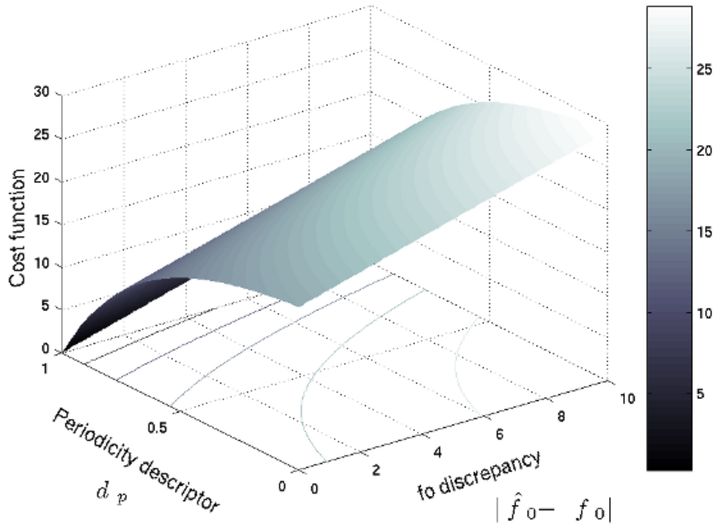


Fig. 3. Graph of the cost function defined by equation (2) and contour plot highlighting the exponential dependance of the cost function on the periodicity descriptor d_p

3.3 Minimisation of the Cost Function Through Adaptative Simulated Annealing

Generic Simulated Annealing. Simulated annealing technique (SA in the following) has been developped to statistically find the best global fit of a non-linear constrained non-convex cost function. This method is derived from the Metropolis method ([17]). Using notations introduced in section 3.2, the aim is to find the global minimum of $\mathcal{C}\left(\hat{f}_0, f_0\right)$ defined by equation (2). The principle of SA is recalled below (see for example [18] p444 for details):

1. Choice (random or not) of an initial candidate P_0 leading to a sound frequency f_{0_0} .
2. Choice of an initial temperature[8] T_0
3. While temperature $T_l > T_{\text{end}}$,

[8] For historical reasons, the scheduling in SA is based on the analogy with temperature cooling.

   - Do the following steps m times
     - Generate randomly a new candidate $\tilde{P}_{k+1}$ (sound frequency $\tilde{f}_{0_{k+1}}$), neighboor of P_k (sound frequency f_{0_k}) according to the probability density g_{T_k} (practically the neighborhood area decreases with T_k)
     - Calculate $\Delta\mathcal{C}\left(\hat{f}_0, f_{0_k}, \tilde{f}_{0_{k+1}}\right) \triangleq \mathcal{C}\left(\hat{f}_0, \tilde{f}_{0_{k+1}}\right) - \mathcal{C}\left(\hat{f}_0, f_{0_k}\right)$
     - If $\Delta\mathcal{C}\left(\hat{f}_0, f_{0_k}, \tilde{f}_{0_{k+1}}\right) \leq 0$, $\tilde{P}_{k+1}$ is automatically accepted as P_{k+1}
     - If $\Delta\mathcal{C}\left(\hat{f}_0, f_{0_k}, \tilde{f}_{0_{k+1}}\right) > 0$, $\tilde{P}_{k+1}$ is accepted as P_{k+1} with the probability $h_{T_k}\left(\Delta\mathcal{C}\left(\hat{f}_0, f_{0_k}, \tilde{f}_{0_{k+1}}\right)\right)$. Otherwise $P_{k+1} = P_k$.
     - $k \leftarrow k+1$
   - Annealing schedule : decrease temperarure ($T_{l+1} < T_l$) according to the chosen cooling law
4. The last state P_{end} is statistically the best fit, i.e. $\mathcal{C}\left(\hat{f}_0, f_{0_{end}}\right)$ is the global minimum of function $\mathcal{C}$.

Extension to an Adaptative Algorithm. Since computation time is often a limiting factor for probabilistic search of a global fit, we decided to take advantage of a refined SA algorithm developped by L. Ingberg and called VFR (Very Fast Re-annealing, [19]). This algorithm introduces adaptative (or re-annealing) capabilities to allow an automatic adaptation to changing sensitivities in the parameter space. Moreover it provides an annealing schedule for temperature T decreasing exponentially in annealing time k. As explained in [19], this is faster than more classical annealing schedules such as the Cauchy annealing (corresponding to the scheme $T(k) = T_0/k$) and much faster than the Boltzmann annealing (where $T(k) = T_0/\ln k$).

3.4 Results and Discussion

Optimization results: The trumpet/trombone models have been tuned on three octaves. For a particular note, around thousand candidates have to be generated by the VFR process. The range of research for each parameter (i.e. the extrema values allowed) didn't appear to be of critical importance. Therefore, large ranges were generally used, so that they could be kept unchanged for many adjacent notes. The quality of the result given by the optimisation process is evaluated through real time playing of the model. If needed, the optimization process is launched again while advantaging the exploration of the parameter space against the rapidity of convergence. This is done by increasing the number of candidatesgenerated at each temperature step (parameter m in the algorithm described in section 3.3).

Implementation / Computation time: Practically, the flowchart described in figure 2 implies code coupling, between the model itself, the f_0 calculus and the VFR process. This can be very time consuming if the computation time is not kept in mind as a priority. Therefore the three standalone codes have been melt into a single code (function calls prefered instead of data files exchange). The tuning of a note lasts about 13 minutes (for 1000 iterations) on a low-end computer (Athlon XP 1800^+ running at 1533Mz with 768Mo of memory).

Ease of use: The optimization process presented above is far from being straigth-forward to use. In fact, several parameters of VFR have to be adjusted (initial temperature, number of parameters generated at each temperature, parameters of the cooling law . . .), which requires some experience and is definitely problem-dependent.

Current research: Different notes produced by the model (tuned with the f_0 -based optimization) may sound with timbre discrepancies. This is not very surprising since many parameters combination lead to the same sound frequency (but not necessarily with the same timbre). Therefore, current research is focussed on including timbre descriptors in the optimization. Flowchart presented in figure 2 remains unchanged, but instead of the only fundamental frequency, other timbre descriptors are considered ([20]). This is expected to provide a way of having the parameters tuned so that the model reproduces at best, not only a target f_0 , but a target sound (possibly played by a real soloist).

4 Other Improvements of the Playability of the Model

Physical models of wind instruments are preferably played in real time through breath controllers. The use of a keyboard as a playing interface generate additional limitations that may harm the playability. To adress this issue, a layer of control is added between the player and the model inputs. Thus, direct controls of the player are used to parameterize higher-level precompiled gestures for several playing modes (listed below).

Attack: At noteon, the evolution in time of the mouth pressure p_m is imposed and parameterized by the MIDI velocity. A very simple model of variation (attack/decay/sustain) has been chosen. A similar evolution of parameter ϵ may also be imposed during the attack.

Vibrato: The lips tension is imposed (through the tuning parameter ω_l) by the mapping. However a vibrato may be generated by a modulation of ω_l. The amplitude of the modulation is zero during the attack, and after a delay starts to increase with time. Note that an additional modulation of noise amount (parameter ϵ) has also been retained.

Legato: When a key is pressed while at least one other key has not been released, it is supposed that the player wants the model to play legato. The time of transition between the two notes is made dependent on the MIDI velocity of the last note played. The modelling of the transition itself (change in valve position or not depending on the two notes) is described in [4].

Humanization: In order to avoid long-lasting notes to sound unnatural (because of fixed parameters), "humanization" is introduced for some parameters (typically ω_l and p_m) as random fluctuations around values given by the mapping or the player. Exagerated fluctations can be used to evoke the playing technique of a beginer.

5 Conclusion

The work presented here is part of the Brass project whose aim is to propose to keyboardists virtual trumpets, trombones and saxophones based on physical modelling. This paper focusses on playability enhancement when playing virtual brass instruments through a keyboard. To summarize, from a pragmatic point of view, the issues adressed , we could say that attention was paid to improve attacks punch, automatic tuning of the model, and the "lively" character of the sound produced. The following sound examples[9,10] highlight these efforts, since they are played live on a MIDI keyboard (without any post-modification of the MIDI inputs).

We think that studying gestures of real brass players would be of high interest to control physical models and to achieve better sound synthesis results. This will hopefully be done in near future.

This paper is focussed on the playability aspects of the virtual instruments. Therefore many aspects of the project are not even mentionned in this paper. The resulting software will be available around December 2005 and will be demonstrated at the conference. Four virtual instruments are expected to play at the same time on recent personal computers (Apple or PC).

Sound examples can be heard on the commercial webpage for this project `http://www.arturia.com/en/brass/brass.php`.

Acknowledgements

The authors would like to thank Xavier Rodet and Axel Röbel for providing the code developped in [16], and Lester Ingber for distributing the VFR as a free software package called ASA (`http://www.ingber.com/#ASA-CODE`). The Brass project was funded by the RIAM program (http://www.riam.org/riam/).

References

1. J.O. Smith. Virtual acoustic musical instruments : Review and update. *Journal of New Music Research*, 33(3):283–304, 2004.
2. V. Välimäki. Physics based modeling of musical instruments. *Acta Acustica united with Acustica*, 4(611-617), 2004.

[9] `http://recherche.ircam.fr/equipes/analyse-synthese/windset/brass_ensemble.mp3`
`http://recherche.ircam.fr/equipes/analyse-synthese/windset/Datrump.mp3`

[10] Composed and played by Christian Laffite, sound designer (commercial link: `http://www.dasample.com`).

3. M. E. McIntyre, R. T. Schumacher, and J. Woodhouse. On the oscillations of musical instruments. *J. Acoust. Soc. Amer.*, 74:1325–1345, 1983.
4. C. Vergez and X. Rodet. Comparison of real trumpet playing, latex model of lips and computer model. In *Procedings ICMC'97*, pages 180–187, Thessalonike, September 1997.
5. C. Vergez and X. Rodet. Trumpet and trumpet player : physical modeling in a musical context. In *Proceedings of the International Congress of Acoustics (ICA)*, page CDROM nIV, Rome, 2001.
6. C. Vergez and X. Rodet. New algorithm for nonlinear propagation of a sound wave. application to a physical model of a trumpet. *Journal of Signal Processing*, 4(1):79–87, January 2000. Special issue on nonlinear signal processing.
7. C. Vergez and X. Rodet. Air flow related improvements for basic physical models of brass instruments. In *Proceedings of ICMC'2000*, Berlin, German, August 2000.
8. S. J. Elliott and J. M. Bowsher. Regeneration in Brass Wind Instruments. *Journal of Sound and Vibration*, 83(2):181–217, 1982.
9. A. Almeida, C. Vergez, R. Caussé, and X. Rodet. Physical model of an oboe: comparison with experiments. In *International Symposium on Musical Acoustics*, pages 112–115, Nara, Japan, Avril 2004.
10. K. Ishizaka and J.L. Flanagan. Synthesis of voiced sounds from a two-mass model of the vocal cords. Technical report, Bell. Syst. Techn. J., 1972.
11. H. Bailliet. Modelling of the French Horn-Player system. Master's thesis, INSA Toulouse, September 1994.
12. W. J. Strong. Computer Simulation of a Trumpet. *J. Acoust. Soc. Amer.*, Suppl. 1(87):S138, 1990.
13. X. Rodet, P. Depalle, G. Fleury, and F.Lazarus. Modèles de signaux et modèles physiques d'instruments: études et comparaisons. In *Actes du Colloque Modèles Physiques de Grenoble*, 1990.
14. S. Adachi and M. Sato. Trumpet sound simulation using a two-dimensional lip vibration model. *J. Acoust. Soc. Amer.*, 99(2):1200–1209, February 1996.
15. H. Bouasse. *Instruments à vent.* Delgrave, 1929. ré-édité par Blanchard (Paris, 1986).
16. C. Yeh, A Röbel, and X. Rodet. Multiple fundamental frequency estimation of polyphonic music signals. In *Proceedings of ICASSP*, pages 225–228 (Vol. III), 2005.
17. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem.Phys.*, 21(6):1087–1092, 1953.
18. William H. Press, Saul A. Teukolsky, Xilliam T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 2 edition, 1992.
19. L. Ingber. Very fast simulated re-annealing. *J. Math. Comput. Modelling*, 12: 967–973, 1989.
20. G. Peeters, S. Mac Adams, and P. Herrera. Instrument sound description in the context of mpeg-7. In *Proceeding of ICMC*, Berlin, Germany, August 27^{th}, September 1^{st} 2000.

The pureCMusic (pCM++) Framework as Open-Source Music Language

Leonello Tarabella

computerART project of ISTI/CNR,
Research Area of CNR, Pisa,
via Moruzzi 1 – 56124 Pisa, Italy
leonello.tarabella@isti.cnr.it
http://tarabella.isti.cnr.it

Abstract. The pureCMusic (pCM++) framework gives the possibility to write a piece of music in terms of an algorithmic-composition-based program -also controlled by data streaming from external devices for giving expressiveness in electro-acoustic music performances- and of synthesis algorithms. Everything is written following the C language syntax and compiled into machine code that runs at CPU speed. The framework provides a number of predefined functions for sound processing, for generating complex events and for managing external data coming from standard Midi controllers and/or other special gesture interfaces. I'm going to propose pCM++ as open-source code.

1 Introduction

In order to put at work the mapping paradigm and the expressive facilities offered by the gesture interfaces we realized at *computerART project of CNR-Pisa* for composing and for performing with expressiveness interactive computer music [3], [5], [6], [7], [8], [9], [13] I started to write a very basic library of functions for sound processing. In the long run the library became a very efficient, stable and powerful framework based on pure C programming, that is *pure-C-Music* or *pCM.*

This programming framework gives the possibility to write a piece of music in terms of synthesis algorithms, score and management of data streaming from external interfaces. The pCM framework falls into the category of the *embedded music languages* and has been implemented using the Xcode C language compiler i.e. the native, free, development environment available on the MacOS X operating system of Macintosh computers.

As a result a pCM composition consists of a XCode project assembled with all the necessary libraries able to implement in realtime the typical synthesis and processing elements such as oscillators, envelope shapers, filters, delays, reverbs, etc. The composition itself is a C program that mainly consists of the Score and Orchestra parts.

Everything here is compiled into machine code and runs at CPU speed.

The Object Oriented paradigm is used for defining instruments in terms of class declaration then instanced as many times as wanted. It's a good practice to collect in the Orchestra section all the necessary instruments and to add together the result of the computation of every instrument is use.

For that the name of the framework is pCM++. At the end, pCM++ is a realtime programming music language that in respect to the other textual languages such as CSound [1] and SuperCollider [4], has two main advantages: there is no need to learn a *new* language (just because *this is* the C language itself) and what's more, it is much faster (more than 10 times) in respect to those quoted because pCM++ entails compilation rather than interpretation.

1.1 Past and Future

I have already presented previous implementations of pCM at other conferences [10], [11], [12]. The new version here introduced runs as host of Xcode compiler (MacOSX). It is easily portable to other platforms and also provides a general purpose console interface the composer can use for setting up synthesis algorithms, for monitoring in/out audio signals and MIDI messages. Printout alerts and others basic interaction features (not oriented to live performance) are also provided.

The pCM++ framework has been written in Standard C++ [2] and makes use of the PortAudio and PortMIDI libraries commonly used as open-source and multi platform libraries able to manage audio signals and MIDI messages in real time. As a consequence it can be easily ported to other platforms such as Windows and Linux.

My aim is towards releasing pCM++ as open source code to be used and enriched by other composers and researchers.

2 Composition

A pCM++ composition consists of an Xcode-C-compiler project properly packed using all the necessary standard C libraries and the new original library that puts the framework at work. This library consists of a number of functions called *elements* (at the moment more than 50) able to implement in realtime the typical synthesis and processing building blocks such as oscillators, envelope shapers, filters, delays, reverbs, etc.

A composition is organized as an usual C program where the main program opens and closes all the necessary audio and MIDI message channels and calls in sequence the movements that make part of the composition. The default name of the main program is `Score` and is intended as an algorithm (from simple to complex) which also may include sequences of note-events as requested by scored music.

The `Orchestra` that generates the audio signal is defined as a separate void and consists of two main parts: the declaration of the elements (such as oscillators and filters) used for defining the synthesis algorithm, and the synthesis algorithm itself.

The orchestra, as expected, usually consists of more than one instrument.

The Score is the program part, which triggers and *feeds* the instruments by assigning proper values to common variables.

Here and in the following, the `courier font` is used for example programs and **`bold names`** are used to indicate reserved keywords of pCM. This is an example of a very simple composition:

```
float  vol,freq; //common variables for communication
                 //between Score and Orchestra

DefOrchestra  Ther() {  //name of the orchestra
  sinosc sinT=newOsc(); //declaration of an oscillator
  BeginOrch
   float alfa=vol*Sinosc(sinT,freq);
   outLR(alfa,alfa);
  EndOrch
}
void Score() {
  float hor,vert;
  InitpCM
  Orchestra=Ther; // The Ther orchestra is activated
  Movement
     GetMouse(&hor,&vert);
           freq=100.+hor*1000.;
     vol=vert;
  EndMovement
  ClosepCM
}
```

An Orchestra is identified by a name and referenced when necessary. All the necessary variables are defined following the C language syntax. Values are assigned to variables by instructions that make part of the program defined in the Score section, that is the composition, or by data coming from the external. The Orchestra uses common variables for getting parametric values computed by the Score.

The synthesis algorithm consists of the code placed in the block `BeginOrch-EndOrch` that also includes the `outLR(.,.)` element. It is possible to define as many as wanted orchestras but only one at a time can be active. The Score activates an orchestra with the instruction `Orchestra=orchname.`

The `Score` is the composition: it initializes pCM, activates the current orchestra, open (if necessary) MIDI channel, etc.; usually a composition is made up of a number of movements each defined by the block **`Movement-EndMovement`**. The block is intended as loop that exits under certain conditions (here not examined).

The two functions run as two concurrent processes at different rates. The Orchestra function is automatically (*by a callback mechanism*) called at each audio buffer switch and since an audio buffer of 256 samples long and a 44100kHz sampling rate are used, the Orchestra function is called every 5.8 msec. The rate of the `Movement` depends on the computer performance and on the complexity of the composition. Then it is not predictable but, as experienced, it is fast enough to complete all the realtime functionalities as required.

3 Toolkits

A composition makes use of the functions belonging to the original library which puts the pCM framework at work at both Score and Orchestra levels. This library consists of three different groups of functions named *toolkits*, each one devoted to specific tasks: DSP, Events and Command toolkits.

1. DSP toolkit deals with the synthesis and processing of sound and groups elements such as oscillators, envelope shapers, filters, delay lines, reverbs, etc. (Orchestra level).
2. Events toolkit deals with the generation and the scheduling of events including timing and management of external events (Score level).
3. Commands toolkit manages messages coming from and sent to the Midi interface, controls the activation of the computer built-in CD player, allows to directly record onto memory the audio signal and to store it onto disk as .aiff or .wave file, provides miscellaneous mathematical functions (Score level).

Each toolkit gathers together functions of the same kind and shares with the others groups the same approach for defining and using the basic pCM elements. An *element* is a building block such as an oscillator, a delay line, a filter, an envelope, etc., used for assembling synthesis algorithms; pCM elements are declared in the same way ordinary C variables (e.g. `int, long, float, bool`) are declared. This is the list of the types provided, so far, by pCM:

```
oscillator, pluck, noise, pulse, envelope, slider, delay-
line, lpfilter, hpfilter, bpfilter, reverb, delay, sample,
scheduler…
```

All types are `low-case` keywords and the elements are usually defined at the beginning of an Orchestra definition and of the Score. What follows is a quick overview of the main elements belonging to the pCM-toolkits as provided so far.

3.1 DSP Toolkit

This is the main toolkit that consists of those functions, which generate and process the audio signal therefore used inside an Orchestra.

Remarks. Computation of audio signals is performed in floating point mathematics so that variables and parameters must be defined as float. Values are normalized at 1.0 so that oscillators range between -1.0 and 1.0, envelopes range between 0.0 and 1.0, the overall signal sent DACs ranges between -1.0 and 1.0.

Time and durations are given in seconds.

§ An oscillator is defined with the type

`oscillator` `myosc=`**`newOsc`**`(phase);`
and used in **`Orchestra`** as follows:
`val=`**`Osc`**`(myosc,freq);`

The **`Osc`**`(myosc,freq)` function returns the proper value for oscillator `myosc` at each sampling tick and stores it into the `val` variable (previously defined). The first parameter of **`Osc`** identifies the oscillator by the name given when defined; the second parameter specifies the current frequency of the oscillator, that can be given as a numeric value, a variable or a complex expression. The phase of the oscillator can be initialized when defined.

§ In order to create a delay line, the **`delayline`** type is used. The instruction

```
delayline mydelay=newDelay(12.5);
```

defines a delay line identified with the name `mydelay` and allocates a memory space corresponding to a duration of 12.5 seconds. There exist two main functions for using a delay line with the expected way of working:

`PutDelay(delayname,value);` and
`val=GetDelay(delayname);`

§ An envelope requires more information, necessary for describing its shape in terms of breakpoints: the *enum* type facility of C is used for this task by defining a list of numbers arranged as the number-of-break-points followed by the sequence of break points given as couples of values (time,value). Then the envelope is created in this way:

```
float  ev={3, .0,.0, .5,1.0, 1.2,.0);
envelope myenvel=newlinEnv(ev);
```

The **`newlinEnv`**`(ev)` function allocates the necessary memory, computes all the values for the envelope by linearly interpolating the break points and returns the pointer to the envelope then stored in `myenvel`. It's also possible to use exponential interpolation using **`newexpEnv(.)`**; in this case break-points must given in triplets, third value referring to the mode how exponential curves must be computed.

The main functions which manage envelopes are **`trigEnv`**`(envname)` used at `Score` level, which starts the scanning of the envelope and **`Env`**`(envname)` used in `Orchestra` that returns the current value of the envelope.

§ Sound samples are defined and loaded with

```
sample mysample=LoadSample(filename.wav);
```

As seen for envelopes, there exist two main functions that manage samples: **`trigSample`**`(sampname);` which starts the scanning of the sample and used at `Score` level, and **`Sample`**`(sampname);` which returns the current value of the sample (used in `Orchestra`).

§ Filters are defined with no particular specification but their behaviors: lowpass, highpass, bandpass, resonator, etc.

```
lpfilter  mylpfilt=newlpFilter();
hpfilter  myhpfilt=newhpFilter();...
```

In `Orchestra` they are used as follows

```
va=LPFilter(mylpfilt,signal,stopbandl);
vb=HPFilter(myhpfilt,signal,stopbandh);
```

§ A reverb is defined with **`reverb`** `myrev=`**`newReverb`**`();`

and used in Orchestra as expected:

```
currval=Reverb(myrev,signal);
```

§ A further element makes part of the DSP toolkit which is not precisely involved in sound synthesis or sound processing but it is a rather useful tool for overall controls: this is **Fader**(fname). A fader is created and initialized as follows:

```
fader  myfader=newFader(initval);
```
Then it is setup with wanted values by
```
setFader(myfader, when, dur, start, end);
```
and used with `currVal=Fader(myfader);`

The value of *when* is the precise moment (see 3.2 paragraph) when the fader is planned to start moving; *dur* tells how much time the fader takes for going from `start` to *end.* A fader is a general purpose facility that can be used, for instance, -for controlling the general volume of an instrument or of the whole orchestra, -for gradually introducing the amount of a sound effect such as chorus, flanging, reverb, etc.. A fader can be used at both Score and at Orchestra levels.

Final comment. What here reported is only a small but significative excerpt out of the DSP Toolkits developed so far: -there exist other generators such **Pluck** (based on the Karplus-Strong algorithm), **Pulse**, **Noise**, **Addosc** (based on a predefined harmonic-based look-up-table): -envelopes can be defined also as Attack-Sustain-Release envelopes and the **Envasr**(envnm) and **ReleaseEnvasr**(envnm) must be used; -audio samples can be played with different speeds using **Sample-speed**(sampname,speed); -delay lines may be read inside the line with **TapDelay**(gap); -two other fundamental functions make part of this Toolkit which perform signal transmission to DACs and signal input from ADCs: **outLR**(left,right) and **inLR**(&left,&right).

3.2 Events Toolkit

The pCM framework has been implemented mainly bearing in mind the algorithmic approach to composition and the interactive gesture controlled live performance of electro-acoustic music paradigm. That entails the program that describes the composition issues events in terms of timed sets of values, which affect the instruments defined in the Orchestra. Since everything happens in real time under the control of the running program/composition, events are treated with reference to the global variable `Time` that holds the updated realtime clock value. The `Resettime` directive forces `Time` to 0 so that it reports (in seconds) the time elapsed since the last `Resettime`. Testing `Time` can generate events such as a note-triggering. However there exists a much formal and efficient approach for managing events consisting of the so called *Scheduler mechanism.*

A Scheduler, in the pCM framework, is an element which gives the possibility of enqueueing timed events in order to be taken into consideration later at the *right* time. As usual, it's necessary first to define a Scheduler element:

```
scheduler mysched=newScheduler(evnum);
```

The **Event**(schedname,dur,value) function enqueues the event defined as duration-value couple, into the specified scheduler. This function is usually invoked at Score level and can also be affected by data coming from the external. Once the events are placed in the Scheduler queue, the instruction

```
if(nextEvent(schedname,&retval)) do_something(retval);
```

is used for checking whether or not the time duration of the current event is finished. If yes, **`nextEvent`** returns *`true`* and `retval` has a valid value of the next event that will be used in the instruction *`do_something`* that usually trigs an instrument.

`GetMidi``(&cmd,&chn,&val1,&val2);` is a boolean function which returns *`false`* if no MIDI message has been received; otherwise it returns *`true`* and the `cmd, chn, val1, val2` variables report the proper values.

3.3 Commands Toolkits

This toolkit groups those functions that work as commands and directives for the composition. Their names specify what they do. The following four commands are usually placed in the main program: **`AudioOpen`**, **`MidiOpen`**, **`Midi-Close`**, **`AudioClose`**. **`ClickToStart`** and **`ResetTimer`** are usually placed, where requested, in the movements. The following commands allow to play and to control audio tracks from the CD driver:

```
CDtrackSearch(tracknum);
CDplay and CDstop
CDvolume(vol);
```

`Record``("filename");` starts to record the global audio signal (as sent to the DACs) onto memory with no loss of quality and save it as a file onto the Harddisk with the specified name and .aiff or .wav extension format.

3.4 Expandability

Since toolkits are defined as collections of ordinary C language functions and are clearly visible by the user, they can be upgraded as wanted and/or requested. It's a good practice, however, to follow the same approach I used in developing the current state of the three toolkits in order to be consistent with all that already existent.

4 Instruments as Objects

The `Orchestra` computes the audio signal by processing the instructions which implements the instruments as defined by the composer using the DSP toolkit functions.

More than one instrument can be defined inside an orchestra, the number depending on their complexity and the power of the computer in use. In any case, when many instruments are defined in an Orchestra, conflict problems coming from the names of variables, delay lines, filters and envelopes in use may arise, specially when putting together previously programmed instruments.

In order to avoid these problems the object-programming paradigm has been introduced for defining and using instruments. Besides, as a result, a cleaner layout for the program-composition is reached.

An instrument is then defined as a class object and declared, that is, instanced in the movement as many times as required. This is done using the very formal criteria of Object Oriented programming. The following is an example of a simple instrument based on the `pluck` element with some other additional elements that enrich its functionality:

```
class    Stringks {
         private:    pluck     string;
                     envelope  envks;
                     envelope  envlpf;
                     delay     rit;
                     lpfilter  filtks;
                     bool      created;
                     float     vala,vax,vaxrit;
                     float     valf,pitch;
         public:     Stringks();
                     void      trig(float freq);
                     float     tick();
                     virtual   ~Stringks();
};
```

The Stringks class is defined as public object that includes both the private section where the elements and the variables are defined and the public section where the methods are declared. Usually, in the class instrument declaration, three are the methods declared plus one for destroying the instanced objects. These methods do the following jobs: -the constructor that actually creates the object and does everything necessary in order to the object works properly such as to create delay lines and envelopes, to load samples, etc.; -activates (`trig`) the synthesis algorithm and, finally, -performs (`tick`) the synthesis algorithm which actually computes the signal. Then the methods are given.

```
void  Stringks::Stringks () {
      string =newPluck();
      rit    =newDelay(0.05);
      filtks =newLPFilter();
      float  st[]={3,.0,.0,1.2,.01,1.0,1.5, 2.,.0,.0};
      float  fl[]={3,.0,.0,1.2,.01,1.0,1.5, 2.0,0.0,0.0};
      envks  =newexpEnv(st);
      envlpf =newexpEnv(fl);
}
void  Stringks::trig(float frq) {
      pitch=frq;
      trigPluck(string,pitch);
      trigEnv(envks);
}
float Stringks::tick(){
      vax=Env(envks)*Pluck(string,pitch)+getDelay(rit);
      putDelay(rit,vax);
      vala=LPFilter(filtks,vax,(Env(envlpf)*1000));
      return vala;
}
```

Remarks. Basically, sound is here generated using the well known Karplus-Strong algorithm; however it is also controlled by an envelope shaper for better control on the sound *coda*; the signal is also added to a delayed copy of itself and then filtered with a variable-cut-frequency lowpass filter. It's now possible to define an instrument for an `orchestra` the same way that the primitive pCM elements are declared. The only difference consists in putting a * pointer before the name which identifies the instantiated object as required by Object Oriented programming:

```
Stringks *mystring=new Stringks;
```

It's worthwhile to invoke soon the `setup` method in order to initialize the just created instrument. From now on, the `trig` method (with the wanted frequency given as parameter instead of through common variables) will be used at `Score` level and the `tick` method will be used at `Orchestra` level for getting the computed signal.

In the following Score example some of the features previously introduced are used: a `scheduler` for composing a semitone scale and two `faders`, the first one for rising the global volume and the second one for panning sound from left to right. The instrument `corda` is defined as an instance of the `Stringks` class just defined. Delay and panning effects are treated at orchestra level rather than in then object-instrument: it's up the composer to put here effects or to include them in the instrument definition. Notes are issued using **`nextEvent(.,.,.)`**.

```
Stringks *corda=new Stringks;
DefOrchestra KSorch() {
   BeginOrch
       val    = Fader(fadin)*corda->tick();
       signR = val*Fader(panpot);
       signL = val*(1.-Fader(panpot));
       outLR(signL,signR);
   EndOrch
}
void  Score(); {
  float      note,semitone;
  scheduler  scale(15);
  fader      fadin;
  fader      panpot;

  Orchesta=KSorch;
  for(int k=0,semitone=440.;k<=12;k++)
     {Event(scale,0.5,semitone);semitone=semitone*1.059;}

  InitpCM
  ResetTime
  Movement
      StartFader(fadin, 0.,2.,0.,1.);
      StartFader(panpot,0.,6.,0.,1.);
      if(nextEvent(scale,&note)) corda->trig(note);
  EndMovement
  ClosepCM
}
```

The composition is now ready and when Score is activated a scale from A3 to A4 with the string timbre lowpass-filtered and panning from left to right, is generated.

5 Conclusion

The pCM framework has been efficiently used for composing and performing many pieces of music under the control of the gesture tracking systems and devices realized at computerART project at ISTI/CNR in Pisa. The pCM framework has been implemented first for Macintosh computers. The current version here reported is the result of the experience I gained using the previous versions. As declared, I'm going to port the work for other platform and put it on the Net as a freeware open-source code music language.

Acknowledgements

Special thanks are due to Massimo Magrini who greatly contributed to set up the pCM main mechanism currently in use and the many facilities for data communication and audio processing. Also thanks to the many students who who contributed to improve and/or optimize the pCM Toolkits. pCM has been also included as special topic in the course of Computer Music I yearly teach at the Computer Science Faculty of Pisa University.

This work is partially supported by the EU Project MUSCLE-NoE (FP6-507752).

References

1. Boulanger, R.: The Csound Book, The MIT Press, (1999)
2. Cargil, T.: C++ Programming Style, Addison-Wesley Professional Computing Series, Readings, MA, USA, (1992)
3. DePoli, G.: Expressiveness in music performance: Analysis and Modelling. In Proceedings of SMAC03, (Stockholm Music Acoustic Conference) Stockholm, (2003)17-20
4. McCartney, J.: http://supercollider.sourceforge.net/
5. Paradiso, J.A.: New way to play; Electronic Music Interfaces. IEEE Spectrum 34-12, (1997) 18-30
6. Rowe, R.: Machine Musicianship. Cambridge: MIT Press. March, ISBN 0-262-18296-8, (2001) (343-35)3
7. Tarabella L., Magrini M., Scapellato G.: Devices for interactive computer music and computer graphics performances, in Proceedings of the IEEE First Workshop on Multimedia Signal Processing, Princeton, NJ, IEEE cat.n.97TH8256, Computer Society Press, (1997)
8. Tarabella L., Bertini G.,: Giving expression to multimedia performances. In Proceedings of ACM Multimedia 2000, Workshop “Bridging the Gap: Bringing Together New Media Artists and Multimedia Technologists” Los Angeles. (2000)
9. Tarabella, L., Bertini G.: The mapping paradigm in gesture controlled live computer music. In Proceedings of the 2nd International Conference “Understanding and Creating Music”, Caserta -Seconda Università di Napoli, Facoltà di Scienze Matematiche, Fisiche e Naturali, (2002)

10. Tarabella, L., Bertini G.: The mapping paradigm in gesture controlled live computer music. In Proceedings of the 3rd International Conference “Understanding and Creating Music”, Caserta -Seconda Università di Napoli, Facoltà di Scienze Matematiche, Fisiche e Naturali, (2003)
11. Tarabella, L.: The pCM framework for realtime sound and music generation. In Proceedings of the XIV Colloquium on Musical Informatic (CIM2003) Firenze, Italy, (2003).
12. Tarabella, L.: Improvising Computer Music: an approach. Sound and Music Computing, Ircam,Parigi. At http://smc04.ircam.fr/ (2004)
13. O'Modhrain, S.: New Gestural Control of Computer-Based Musical, In Proceedings of NIME02 (New Interface for Musical Expression), Dublin, (2002)

Timbre Variations as an Attribute of Naturalness in Clarinet Play

Snorre Farner[1], Richard Kronland-Martinet[2], Thierry Voinier[2], and Sølvi Ystad[2]

[1] Department of electronics and telecommunications, NTNU, O.S. Bragstads plass 2B, 7491 Trondheim, Norway
`farner@iet.ntnu.no`

[2] Laboratoire de Mécanique et d'Acoustique, CNRS, 31, chemin Joseph Aiguier, 13402 Marseille Cedex 20, France
`{kronland, voinier, ystad}@lma.cnrs-mrs.fr`

Abstract. A digital clarinet played by a human and timed by a metronome was used to record two playing control parameters, the breath control and the reed displacement, for 20 repeated performances. The regular behaviour of the parameters was extracted by averaging and the fluctuation was quantified by the standard deviation. It was concluded that the movement of the parameters seem to follow rules. When removing the fluctuations of the parameters by averaging over the repetitions, the result sounded less expressive, although it still seemed to be played by a human. The variation in timbre during the play, in particular within a note's duration, was observed and then fixed while the natural temporal envelope was kept. The result seemed unnatural, indicating that the variation of timbre is important for the naturalness.

1 Introduction

Naturalness is a term that is regularly used in the context of speech and music synthesis, although a clear definition is difficult to formulate. Ternström [1] has suggested a layered transport model of communication, be it musical or spoken, where in the present case a musical message at the first layer may be converted to a script of musical phrases in the second layer, then to a sequence of notes with certain attributes, further to gestures (e.g. movements of the bow of a violin) that are finally converted by the instrument to sound waves in a last layer. The sound waves are transmitted to the listener after being distorted by room acoustics and possibly by microphones and loudspeakers. Another important aspect is of course the layer of the listener's perception, but we will not be concerned with this here.

By considering naturalness in this way, one can suppose that a defect in any layer may cause the sound to be perceived as unnatural. As an example, in synthesis of musical instruments, lack of naturalness seems to occur either at the gesture level, for instance when a control interface cannot sufficiently capture the player's gestures, or at the instrument level, due to a poor instrument or radiation model. In general, for the purpose of music and speech synthesis, naturalness may be defined as the attribute that makes the listener think that the sounds are produced by a human. Rodet [2] has referred to this definition in the context of synthesis of the singing voice, and Nusbaum et al. [3] asked their subjects in a listening test whether the vowel they heard was produced by a human or by a computer, thus also adhering to this interpretation of the term naturalness.

An important issue concerning naturalness and sound synthesis is related to the fact that at present a computer cannot mimic the human speech and music performance in a convincing way, unless it copies the performance. Even when a natural sound is distorted during transmission, for instance by the telephone where higher frequencies are lost and noise is added, we do not doubt that the speaker is human. This example shows that an important part of the naturalness, as defined above, is contained in the control of the sound source rather than in the sound itself. We therefore search for cues that are important for rendering a music performance natural, in the sense that the listener thinks that it is performed by a human and not by a computer program.

A number of studies have already been conducted on performance rules defining the performer's deviations from musical scores. The rules may be divided in two main categories [4]: differentiation rules linked to duration, pitch, and intervals, and grouping rules linked to the way the performer gathers tones into melodical gestures, subphrases, and phrases. To our knowledge, no rules have been established that take into account variations in timbre during the notes. This is necessary for self-sustained instruments, such as the clarinet or the violin, as the sound is also controlled after the note onset. Hence, a new family of expressive parameters contributes to the performance and gives new degrees of freedom that act on the naturalness.

In the present study we take a closer look at the variation of such parameters and their relation with the naturalness in the clarinet case. More precisely we divide the naturalness at the gesture and instrument layers into a systematic and a random part. In fact, even a musician who controls his instrument to perfection cannot perfectly reproduce the exact same musical phrase. There are muscle vibrations, small variations in the player's lip position, phase variations of waves in the resonator and much more. It should be mentioned that the GERMS model [5] goes further and divides a performance into five principal expressive components: Generative rules, Emotional expression, Random variations, Motion (i.e. gesture), and Stylistic unexpectedness (hence the acronym). The systematic part should encompass all but the random variations of this model.

We address the systematic and random variations in the control parameters during each note and verify our hypothesis that these parameters follow rules in a similar way as do the variations between notes. We also link the variation in timbre, here represented by the spectral centroid, to the perception of naturalness.

Finally, "[Sound x]" refers to sound examples on the CMMR Internet site: http://www.lma.cnrs-mrs.fr/˜cmmr2005/

2 Sound Preparations

In order to test this hypothesis, it is important to be able to measure the control parameters employed by the musician as well as parameters that the musician does not control, such as room acoustics. Although a real acoustical instrument would be preferred for studies on naturalness, an electronic instrument was chosen since it allows fine measurements of the control parameters as well as reinjection of these parameters into the synthesis model.

The physics of the clarinet has been studied for a long time, and simple relations describe quite realistically its sound production [6, 7, 8]. Without giving a detailed description of all quantities involved, the model can be summarized by three equations simulating the reed motion, the pipe resonances, and the nonlinear interaction between the reed motion and the pipe resonances. The reed motion is modelled as a spring with mass and damping:

$$\mu \frac{d^2y}{dt^2} + r\frac{dy}{dt} + ky(t) = p(t) - P, \tag{1}$$

where P is the mouth pressure, y the oscillating reed displacement from equilibrium, and p the oscillating pressure inside the mouthpiece. The resonances of the pipe may be described by its input impedance

$$\tilde{Z}(\omega) = jZ_c \tan(\omega L/c - j\alpha\sqrt{L}), \tag{2}$$

which relates p and u to the pipe length L. The Bernoulli equation may be used to couple the two equations together by the oscillating volume flow u of air through the mouthpiece:

$$u(t) = w(y(t) + H)\sqrt{\frac{2(P - p(t))}{\rho}}, \tag{3}$$

where H is the equilibrium height of the reed opening. This system of equations can be solved in many ways, but the main point is that the model provides the musician with three control parameters: the length of the pipe L, the blowing pressure P, and the equilibrium reed opening H.

This or a similary physical model is implemented in the Yamaha VL70-m virtual acoustic synthesizer [9] although we do not have access to the true contents and real-time implementation. The synthesizer may be piloted by various controllers, and we have here used the clarinet-like Yamaha WX5 MIDI controller. The fingering determines the note to play and thus the length L of the pipe, the blowing pressure P is captured as the MIDI breath-control parameter BC, and the reed opening H is represented by a MIDI “general control” parameter that we will call the (equilibrium) reed displacement RD. The latter is controlled by the lower lip against a non-oscillating reed. In addition, the reed displacement may also give a pitch bend, i.e. a slight variation of the pitch, but this was disconnected in the present study and is disregarded in the following. The musician’s actions on the WX5 are transmitted by MIDI parameters to the VL70-m, which synthesizes the sound in real time based on a model similar to the one described and depending on the variations in the fingering, the reed displacement, and the blowing pressure.

This setup gives the musician a realistic, albeit limited, control similar to a clarinet and thus offers some expressive control of the timbre of the sound. While the relation between the physical quantities P and H is not directly related to the MIDI parameters BC and RD, and not knowing exactly how they are treated in the synthesizer, we will only consider the MIDI parameters (scaled to range from zero to one) in the following.

A 20-second theme of the melancholic song "Plaisir d'amour" (Jean-Paul Martini 1760) [Sound 1] was played 20 times by an amateur clarinetist, who was asked to play all repetitions as equally as possible, with the same interpretation and expression. A metronome was used to facilitate comparison between the 20 repetitions. The MIDI data (breath control, reed displacement, note-onset timing with note value, and metronome timing) for each performance was saved to a text file, and the synthesized sound was recorded to a WAV file. The data were processed in Matlab.

3 Systematic and Random Variations

Figure 1 shows the MIDI parameters BC and RD for two of the 20 versions played [Sound 2, 3]. The vertical gray lines represent the note onset timings while the vertical dotted lines indicate the metronome ticks. The figure shows that BC and RD vary in a regular way, but with some variations from repetition to repetition. The systematic variation will survive an averaging over all the repetitions, while the random variations can be quantified by the standard deviation. A few precautions are necessary, however. First, all performances must be synchronized, which was facilitated by the use of the metronome. Second, the

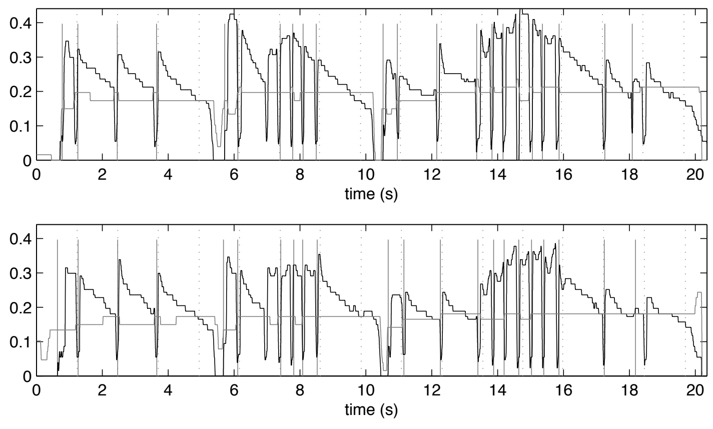


Fig. 1. Variations of the breath control BC (black lines) and the reed displacement RD (gray lines) for two different performances. The vertical lines mark note onset (gray) and metronome (dotted) timings.

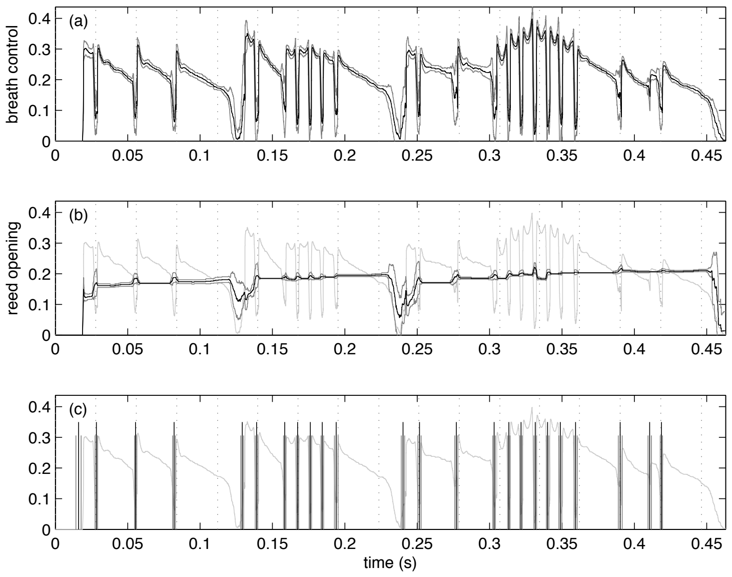


Fig. 2. The systematic and random changes of (a) the breath control BC and (b) the reed displacement RD represented by their normalized mean (black lines) and standard deviation (gray lines). The vertical lines in (c) show the note-on timings (black lines), their standard deviations (gray lines), and the metronome timings (dotted lines).

curves should be normalized note by note so that level variations between the repetitions do not contribute to the standard deviation.

The notes were separated by their note onset timings and shifted to the average timings. Normalization was performed individually for each note: The parameters were divided by the time average for each note, e.g. $\overline{BC}_i = \langle BC_i(t) \rangle$ for the blowing pressure. We ignored values below 25 % of the maximum BC and below 50 % of the maximum RD in order to avoid taking into account pauses and bumps between the notes. Then the normalized curve was averaged over all repetitions $i = 1, \ldots, 20$ and finally remultiplied by the mean of all $\overline{BC}_i$ for this note. The standard deviation was calculated in the same way by normalizing by $\overline{BC}_i$. In this way, the final normalized standard deviation should only include the random variations within each note, i.e. the variations in curve form and not variations in the general level of the notes.

The mean and standard deviation of BC and RD are presented in Figure 2a and b, respectively, while the standard deviation of the note-on timings are shown as vertical gray lines around the solid mean lines in Figure 2c.

The mean of BC shows that there is a systematic variation of the breath control during the play. The long notes have a peak at the start and decrease gradually towards their end. The breath control falls to a minimum for a short moment before the next note is attacked. In the more rapid parts in the middle of the second and third phrases, however, the movement of the notes shows a different shape, especially the short ones: there is also a peak at the end of the note. Additionally, the crescendo in the beginning of the last phrase is manifested by smaller slopes in the long notes and a higher blowing pressure in the middle of the phrase. These characteristics indicate that a set of playing rules may be established. Although a part of the movement is necessary for a sound to be produced, the musician has some liberty to express his intention through manipulation of the control parameter.

This is also the case for the mean of the reed displacement RD, though this parameter to a great degree is kept constant during the notes. The constant reed opening is coherent with the playing technique that is most commonly used among classical musicians. Although the playing style will depend on the construction of the clarinet, changing the reed opening in the model presented in Section 2 makes the brightness of the sound change [10]. In classical music, it is in general desired that the brightness does not vary from note to note, and on the described model, the reed opening may be used to compensate changes in brightness due to variations in the note value and in breath control.

Another characteristic of the mean variation of RD is that there are two deep bumps at the transition between each musical phrase. At these points the player took his breath and thus relaxed the pressure on the reed for a moment. Breathing is a natural part of a clarinet performance and may contribute to the naturalness. The sound of breathing is not included in the synthesis, but it seems evident for the listener where the player breathed. The pause might be sufficient for the perception of natural breathing, but it is possible that the bumps in the reed displacement affect the termination of the previous note and the attack of the following, and thus contribute to the perception of naturalness.

It is interesting to note that playing the averaged parameters by the VL70-m synthesizer gives a result that seems to lack some expressiveness [Sound 4]. However, it may still be considered natural in the sense that a human seems to have played it, although maybe less motivated.

The standard deviation of RD is very small in this study. One reason for this is the steady reed opening, but it is perhaps also due to the discretization induced by the MIDI protocol, as was suggested in Figure 1. Whether the subtle nuances that would result from a better MIDI resolution would be audible has to be investigated in a future work.

4 Timbre Variations

Timbre is defined as the perceptual attribute that distinguishes two tones of equal pitch, loudness and duration [11]. Furthermore, the possibility to *vary* the timbre continuously during the play seems to be important for the musician wishing to interpret the music expressively. It also seems that such continuous variations of the timbre contribute to the naturalness of a performance.

In the following, we look at the variation of the brightness [12] of the sound, which is a commonly used timbre descriptor. It is defined by

$$SCG = \frac{\int_0^{f_c} |\hat{S}(f)| f\,df}{\int_0^{f_c} |\hat{S}(f)|\,df} \tag{4}$$

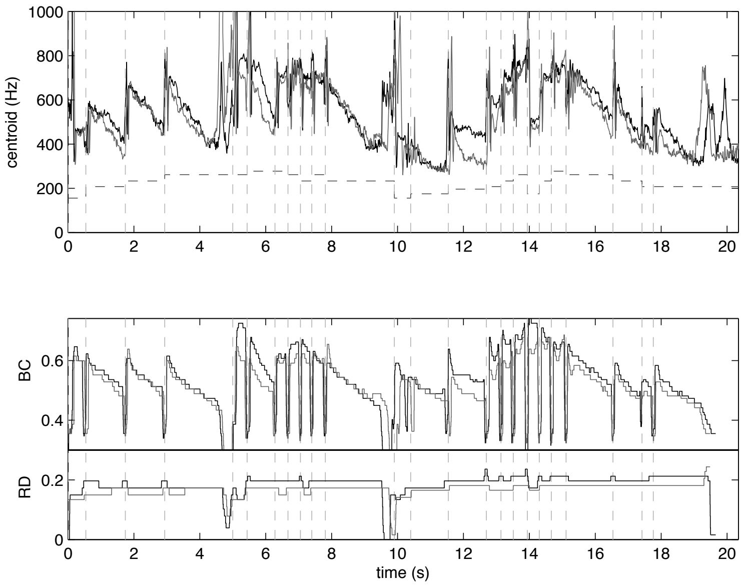


Fig. 3. The spectral centroid of the sound resulting from the two repetitions in Figure 1 (in black and gray) together with the note frequency (dashed line). The corresponding breath control BC and reed displacement RD curves are shown below, and dashed vertical lines show note onsets.

where f and $|\hat{S}(f)|$ respectively represent the frequency and the magnitude of the Fourier transform of the signal $s(t)$. A continuous curve of the centroid frequency was obtained by taking the short-term Fourier transform of 50 % overlapping signal frames of 23 ms, weighted by a Hanning window. The cut-off frequency f_c was introduced because the VL70-m synthesis added noise to simulate blowing noise, which was more pronounced for small blowing pressures. The noise disturbed the calculations of the centroid and was removed by setting $f_c = 2500\,\text{Hz}$. Figure 3 shows the spectral centroid frequency for the two repetitions in Figure 1.

Without the noise, the correlation between the blowing pressure and the centroid is obvious: The sound is brighter in the beginning of the long notes, and darkens as the blowing pressure decreases. This is in accordance with the "Worman rule" that the pressure wave is nearly sinusoidal for low excitation energy (for weak blowing pressure), and that the higher harmonics become more important as the player blows harder [13, 14]. The correlation with the reed displacement RD was ignored in the present study as this parameter was mostly constant during the notes.

The decrease in the centroid frequency was about 300 Hz for the long notes, accompanied by a decrease in energy (not shown here). We have verified by listening that the change in the spectrum from the beginning to the end of these notes was highly audible, even when compensating for the energy decrease. This was done by extracting frames at several points from the beginning to the end of a long note, removing the blowing noise by low-pass filtering, normalizing them to approximately the same loudness, and applying a pitch-synchronous lengthening to make listening possible [Sound 5].

To show the effect of the timbre on the naturalness, we can force the timbre to be constant during the play. This may be done by freezing BC and RD of one of the repetitions to their mean values and feeding them together with the old note-change parameters to the synthesizer. The result sounds static and unnatural, much because also the intensity of this static performance was fixed [Sound 6]. We therefore forced the intensity to vary as for the original performance by multiplying the signal by the envelope of the original. Although it might be accepted that this was played by a human musician, the timbre did not vary in a natural way [Sound 7]. We take this as an indication that variation of the timbre is important for the perception of naturalness.

5 Conclusions and Further Work

The analysis of the 20 repetitions of the 20 s music performance shows that a regular pattern of the two control parameters, breath control (BC) and reed displacement (RD), can be extracted by averaging the performances, while the variations between repetitions, considered the random part, may be quantified by the standard deviation.

For the regular movement of the breath control, qualitative differences was found between notes in the sequences of short notes and the calmer parts of the melody. This suggests that expressive rules may be established for the movement of BC. The reed displacement was found to vary little during each note, at least within the MIDI resolution. This was coherent with the fact that the player had been taught to play with a steady reed.

Because the parameters were normalized, the standard deviation mainly showed the difference in *shape* between the curves of each note. When reconstructing a performance from the averaged parameters, the result seemed to lack some expressiveness compared to any of the 20 real performances. This suggests that the parameter fluctuations quantified by the standard deviation contribute to the expressiveness.

We have verified that variations in the breath control BC induce variations in the timbre, at this stage only quantified by the spectral centroid. When fixing the timbre, but keeping the temporal envelope of the signal, the result becomes unnatural. This indicates that the variation of the timbre is important for the perceived naturalness of the performance.

To elaborate on these preliminary results, we foresee to

– apply a more recent synthesis method [15] that is considered more realistic and where more information on the timbre is available,
– use a professional musician and a larger repertoire of playing styles,
– effectuate a more detailed study of the rules related to the movement of the parameters,
– employ other measures of the timbre, and
– perform listening tests to determine the importance of variation of timbre and intensity as well as timing to the perception of naturalness.

Finally, an interesting question is whether such rules depend on musicians and on playing styles. If so is the case, recognition of musician and playing style from MIDI data could be considered, and it should make it possible to use rules obtained from analysis of MIDI performances to add expression and naturalness to simple score data.

Acknowledgements

This work has been partly supported by the Research Council of Norway.

References

1. Sten Ternström, "Session on naturalness in synthesized speech and music," in *143rd ASA meeting, Pittsburgh*, June 2002.
2. Xavier Rodet, "Synthesis and processing of the singing voice," in *IEEE Benelux Workshop on Model Based Processing and Coding of Audio (MPCA)*, Leuven, Belgium, Nov. 2002, pp. 99–108.
3. Howard C. Nusbaum, Alexander L. Francis, and Anne S. Henly, "Measuring the naturalness of synthetic speech," *International Journal of Speech Technology*, vol. 1, no. 1, pp. 7–19, 1995.
4. Johan Sundberg, "Grouping and differentiation. Two main principles in the performance of music," in *Integrated Human Brain Science: Theory, Method, Application (Music)*, T. Nakada, Ed., pp. 299–314. Elsevier Science, 2000.
5. Patrik N. Juslin, "Five facets of musical expression: a psychologist's perspective on music performance," *Psychology of Music*, vol. 31, no. 3, pp. 273–302, 2003.
6. R. T. Schumacher, "Self-sustained oscillation of the clarinet: an integral equation approach," *Acoustica*, vol. 40, pp. 298–309, 1978.
7. J. Kergomard, "Ch. 6. Elementary considerations on reed-instruments oscillations," in *Mechanics of Musical Instruments, Lecture notes CISM*, A. Hirschberg, J. Kergomard, and G. Weinreich, Eds., pp. 229–290. Springer Verlag, Wien/New York, 1995.
8. J.-P. Dalmont, J. Gilbert, and S. Ollivier, "Nonlinear characteristics of single-reed instruments: Quasistatic volume flow and reed opening measurements," *J. Acoust. Soc. Am.*, vol. 114, pp. 2253–2262, Oct. 2003.
9. R. Rideout, "Yamaha VL1 virtual acoustic synthesizer," *Keyboard Magazine*, vol. 20, pp. 104–118, June 1994.
10. Robin Thomas Helland, "Synthesis models as a tool for timbre studies," Master thesis, Norwegian University of Science and Technology, 2004.
11. ANSI. American National Standard — Psychoacoustical Terminology, (American National Standards Institute, Inc., New York).
12. James W. Beauchamp, "Synthesis by spectral amplitude and "brightness" matching of analyzed musical instrument tones," *Journal of the Audio Engineering Society*, vol. 30, no. 6, pp. 396–406, 1982.
13. W. E. Worman, *Self-sustained nonlinear oscillations of medium amplitude in clarinet-like systems*, Ph.D. thesis, Case Western Reserve University, 1971, Ann Arbor University Microfilms (ref. 71-22869).
14. Arthur H. Benade, *Fundamentals of musical acoustics*, Dover Publication, New York, 2 edition, 1990, 1st ed. published by Oxford University Press in 1976.
15. Ph. Guillemain, J. Kergomard, and Th. Voinier, "Real-time synthesis models of wind instruments based on physical models," in *Proc. of the Stockholm Music Acoustics Conference (SMAC)*, Sweden, Aug. 2003, pp. 389–392, Stockholm.

Scoregram: Displaying Gross Timbre Information from a Score

Rodrigo Segnini and Craig Sapp

Center for Computer Research in Music and Acoustics (CCRMA),
Center for Computer Assisted Research in the Humanities (CCARH),
Stanford University,
660 Lomita Drive, Stanford,
CA 94305, USA
{rsegnini, craig}@ccrma.stanford.edu

Abstract. This paper introduces a visualization technique for music scores using a multi-timescale aggregation that offers at-a-glance information interpretable as the global timbre resulting from a normative performance of a score.

1 Introduction

A musical score using common music notation (CMN) does not convey a literal representation of the sound it notates; rather, it contains the instructions necessary to produce the sound. This realization from score to sound is a convoluted process which may be simplified as follows: (i) pitch and duration are read from the vertically and horizontally positions of symbols on a staff; (ii) associated markings—not always aligned to the symbols they modify—inform us about loudness or articulation-dependent onsets; and finally, (iii) other standard symbols and editorial practices—such as labeling staves with instrument names—complete a minimal description of the sound at a particular moment in the score. By aggregating this information at ever larger time scales this data at the event level becomes integrated into higher level abstractions that serve as the basis for global descriptors like: melodic contour, harmonic complexity, tonality, event density, and intensity, *etc*.

With sufficient knowledge of CMN, one is thus able to derive from these raw graphical symbols a good approximation of how a notated piece sounds as well as its overall form and structure, even without actually performing it.

However, despite the standardization of CMN, various constraints may affect the layout of this data, and our ability to parse it. Space limitations are an example of such constraints, which may force changes in clef, octave markings or in the spacing between symbols, all of which hinder the spatial relationship between a notated event and its audible correlate.

We denominate this kind of mental picture of a score the 'gross timbre information' because it represents the compounded result of the actions by the performer(s) producing the notated sound. This paper introduces an approach for displaying this information directly from the score using computational methods.

1.1 Timbre Information Display

One way to simplify the display of gross timbre information is to use a spectrogram. A spectrogram displays on the vertical axis frequency content in bands of width relative to the sampling resolution—with the amount of energy in a band depicted by grayscale or color values against time on the horizontal axis. The spectrogram's axes are more regularized than a musical score; however, larger musical structures other than the instantaneous surface features are difficult to identify when viewing a spectrogram. Also, spectrograms are limited in their potential to display timbre information due to their emphasis in frequency content rather than more perceptual measures of timbre.

The physical parameters of timbre are usually reduced to a more compact set of features which still describe the acoustical signal, some of them with perceptual relevance. A partial list of such features which can be obtained from the time and/or spectral domain would include: root-mean-square amplitude (power), bandwidth (spread of the spectral energy), centroid (amplitude-weighted average of energy distribution), harmonicity (how much does that energy falls along harmonic partials), density (how much energy per critical band), skew (tilt toward low or high end of the spectrum), roll-off (decay of high frequency partials), flux (between frames), among others.

Grey [2] worked with listeners in similarity experiments so as to determine the perceptual correlate with some of these features, and he produced a timbral space displaying the perceptual distance among notes produced by different instruments. Recent work, as exemplified by Fujinaga [3], Brown [1], and others, uses a host of those features to categorize timbre in an attempt to have computers recognize specific instruments.

1.2 Acoustic v. Symbolic

All of the approaches for timbral description, however, are derived from the acoustic representation of a musical sound, therefore their results are somewhat different from what can be specified by its symbolic representation, namely, the musical score.

Assuming that a score is the closest there is to the original compositional idea, then we have to count every step from there to our ears as potentially transforming factors. There are two major such steps in this path: performers and performance space; performers add vibrato, tremolo, rubato, plus their 'mistakes', and the performance space adds reverberation, and background noise. While many of these factors can be desirable, we sometimes end up with very different acoustic renditions of the same piece. As with listening, whatever structural information that can be derived from this approach becomes biased by the specific performance.

On the other hand, information derived from the symbolic representation is performance agnostic and is a time-honored way of generating gross conceptualizations of timbral content. However, this human-based approach is expertise-dependent and is time-consuming. This presents issues of consistency and speed given variabilities in CMN layouts, but it is very good to obtain information using different time-scales. In other words, humans are able to change their analysis window-lengths ranging from a single time event to the whole duration of the piece. The visualization techniques presented below attempt to keep the advantages of the human-based approach, while dealing with the shortcomings through a computer-based approach.

1.3 Previous Work

Recent visualizations of timbre include *Timbregram* and *Timbrespace* [11]. Timbregram is based on a time domain arrangement of the music (can be superimposed to a waveform display), with colors according to spectral features added to variable-size slices. Timbrespace maps features to objects with different shapes, texture and color in a 2D or 3D virtual space. Their goal is to facilitate browsing of a large number of sound files; the latter also suggests groupings among different pieces. For an experimental study on cognitive associations between auditory and color dimensions see [4].

The most direct predecessor of scoregrams are Craig Sapp's *Keyscapes*, which show tonality structure of a piece. In Keyscapes, the horizontal axis represents time in the score, while the vertical axis represents the duration of an analysis window used to select music for a key-finding algorithm; each analysis window result is shaded according to the output key. Independent analysis group together according to the relative strength of key regions the composition. A more detailed description of the visualization approach is given in [9] and [10].

Scoregrams are also closely related to *Dynagrams* used by Jörg Langer, *et al.*, to study loudness changes on multiple-time resolutions graphs [7]. Both plot axes are similar to keyscapes, but the vertical axis is inverted and the windowing method is slightly different. Dynagrams are used to plot the change in loudness of a recording over time. Crescendos are shown in shades of red, and decrescendos are shown in shades of green. Local dynamic changes display rapid changes in loudness and global dynamic changes can be seen emerging from this low level description of the loudness. Dynamic arches are displayed visually from the interaction of the local and global dynamic descriptions in the plot.

2 Implementation

To introduce the potential of scoregram we will display a single feature from the score—pitch height—according different subdivisions. In these examples, images were automatically generated from CMN data encoded in the Humdrum file format and analyzed using command line programs from the Humdrum Toolkit [6] as well as custom-built programs. Other symbolic representations would be just as good, such as MIDI files.

Meaningful visualizations are accomplished by mapping perceptually relevant features into an equivalent dimension in an alternate domain. Visual elements, for example, have a number of perceptually significant characteristics, such as shape, texture, and color, which can be linked in the auditory domain; some of them, like timbre, are also multidimensional. In this work we mostly explore color which has three perceptual dimensions of hue, saturation, and intensity, and focus on the first of them: hue.

Mapping According to Register. A common association to the concept of timbre in a single instrument is register. The pitch range of most orchestral instruments can be summarily subdivided into three timbral zones each covering about a third of their range (i.e. low, medium, and high). We can determine these thresholds manually (i.e. setting a fixed note value at the boundary), or automatically (i.e.: at the 1/3 and 2/3 percentiles in the events histogram). For the following scoregrams, activity in each gross timbral range is indicated by the colors red, green, and blue, respectively, and it is proportional to the number of tokens from that class in the histogram, normalized by the largest token value of either: (i) all colors across the time-window, (ii) all values of a single color, or (iii) among the three values in that window. Finally, the normalized value becomes a number in the Red-Green-Blue color space. Therefore, a piece with activity only in the mid register would yield a green picture, while simultaneous activity in the extreme registers, would yield magenta resulting from the combination of red (low register) and blue (high register).

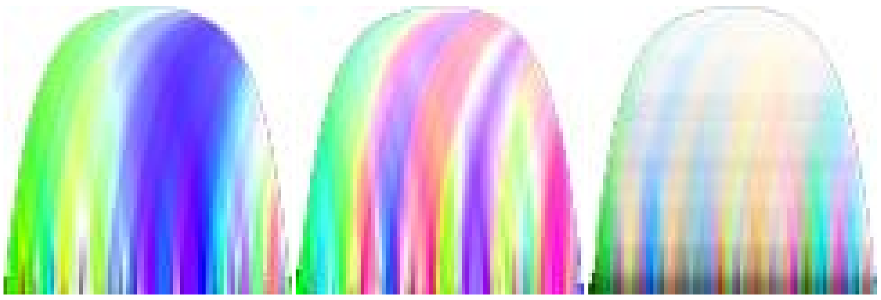


Fig. 1. Three scoregrams using range data. They illustrate a progression from strongly segmented and contrasting range-derived structures to a more more homogeneous structure. These examples are taken from J.S. Bach's fugues (Nos. 14, 1, and 20 from left to right, respectively) in the Well-Tempered Clavier, Book I. No.14 (*left*) has three clear sections where the medium and high registers appear most prominently; No.1 (*middle*) shows more boundaries with no color in particular becoming emphasized; No.20 (*right*) shows all colors with almost equal presence, resulting in an early aggregation toward white at the top of the scoregram.

The images in Figure 1 show at-a-glance aspects about pitch distribution—by extension, register-dependent timbre quality— that are not obvious to the naked eye in a musical score. At the bottom is the event level, quantized to include every 16th-note duration on the score; this is done to keep equal score time for each token. Time goes from left to right, from the beginning to the end. The size of the analysis window increases from bottom to top, so that local features are shown below and global features at the top, which represents the entire duration of the piece. The progression from bottom to top is done in a logarithmic scale to match the way our perception of time works. Each row is the same fraction larger/smaller than the previous row. It can be suggested that the color at the tip of the dome is the characteristic gross timbre of the complete composition.

Another useful piece of information displayed in the scoregram are the color boundaries where register changes occur. For example, the rightmost plot in Figure 1 suggests that the resulting timbre is more uniform since no color becomes emphasized, whereas in the first plot, the movement from mid to high register becomes a distinctive characteristic of the piece.

Other Mappings. Any arbitrary subdivision of the instrumental range is possible. For example, in a microtonal context, fine subdivisions may be necessary to augment the contrast of auditory variations. We have implemented subdivision into octaves—suggested to be a general bandwidth for timbre invariance [5]—and into critical bands for the note pitches (see Figure 2), a more perceptually uniform measure of frequency with a width of about 1/3 octave each; it is generally assumed that timbre can be characterized by the energy contents in each critical band [8]. Since these subdivisions produce more than the three regions which could be conveniently mapped one of the three RGB colors, we used a 2-D interpretation of the color space commonly known as the color wheel, and assigned an angle equivalent to a distinct color wavelength to each one of the 10 (v.g. octaves) or 24 (v.g. critical bands) tokens.

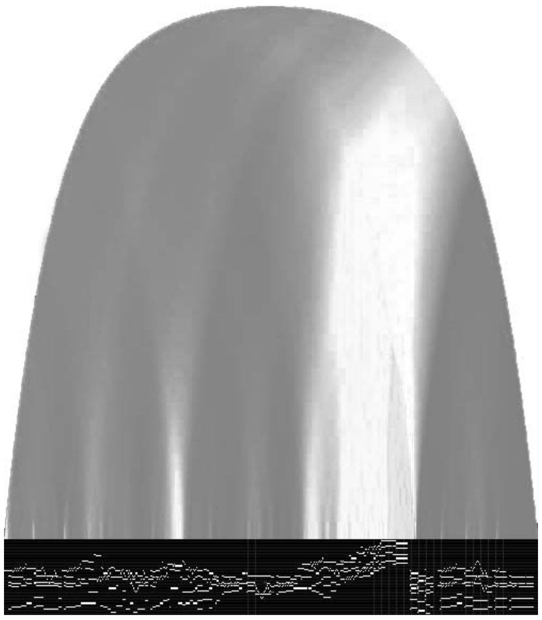


Fig. 2. A scoregram using critical band data from Barber's Adagio for strings. A piano-roll representation is appended to the bottom of the picture to depict the position of musical events. There is a clear boundary at the point were the music reaches a climax in the high register, before returning to the broad low and medium registers.

Figure 2 also demonstrate how more striking structural features will rise higher in the scoregram plot. For example, in this plot the extremely high registration of all instruments about 75% of the way through the piece generate a strong band of contrasting color to the other regions of the piece.

3 Discussion

A scoregram can have various interpretations. For example, a piece whose event distribution is homogeneous across the dimension in which it is measured (e.g. register) may be perceived to be less dramatic than one with marked changes. The idea is that if at the top of the scoregram we can see boundaries preserved from the bottom, or the event-level, it means that the piece has contrasting sections.

Scoregram is extensible to any other types of musical features. We are considering the mapping of multiple features to unused color dimensions. The basic strategy we used is to plot three states in independent RGB values. Interpolating these values in the Hue-Saturation-Intensity (HSI) space can be used to map dynamics, for example, to saturation (*e.g.* how vibrant the color is), and articulation to intensity (*e.g.* how bright the color is).

In the sample of music examined thus far, scoregrams proved useful for detecting basic musical structures based on the musical features being examined. It may also useful for establishing measures of similarity between repertoires and forms, or comparisons between the precisely observable acoustic event and its notated counterpart, which would help to quantify a performer's contribution to the music.

References

1. Brown, J. C.: "Computer identification of musical instruments using pattern recognition with cepstral coefficients as features". Journal of Acoustic Society of America 105 (1999) 1933–1941
2. Grey, J. and G. Gordon: "Perceptual effects of spectral modifications on musical timbres". Journal of the Acoustical Society of America Vol. 63(5) (1978) 1493–1500
3. Fujinaga, I.: "Machine recognition of timbre using steady-state tone of acoustic musical instruments". Proceedings of the international Computer Music Conference (1998) 207-210
4. Giannakis, K. and M. Smith: "Imaging Soundscapes: Identifying Cognitive Associations between Auditory and Visual Dimensions" in Godoy, R. I., Jorgensen, H. (eds.): Musical Imagery. Swets & Zeitlinger (2001) 161–179
5. Handel, S. and M.L. Erickson: "A Rule of Thumb: The Bandwidth for Timbre Invariance Is One Octave". Music Perception 19 (2001) 121–126
6. Huron, D.: "Music Information Processing Using the Humdrum Toolkit: Concepts, Examples, and Lessons". Computer Music Journal 26 (2002) 11–26
7. Langer, J., R. Kopiez, C. Stoffel and M. Wilz. "Real Time Analysis of Dynamic Shaping" in the Proceedings of the 6th International Conference on Music Perception and Cognition, Keele, United Kingdom, August 2000.
8. Moore, B.: An Introduction to the Psychology of Hearing. Academic Press (2003)
9. Sapp, C.: Harmonic Visualizations of Tonal Music. Proceedings of the International Computer Music Conference (2001) 423–430
10. Sapp, C.: "Visual Hierarchical Key Analysis". Association for Computing Machinery: Computers in Entertainment, 3(4) (Fall 2005).
11. Tzanetakis, G.: Manipulation, Analysis, and Retrieval Systems for Audio Signals. Ph.D. Dissertation. Princeton University (2002)

A Possible Model for Predicting Listeners' Emotional Engagement

Roberto Dillon

Nanyang Technological University,
School of Computer Engineering – gameLAB,
50 Nanyang Avenue, Blk N4-B1b-13b,
Singapore 639798
RDillon@ntu.edu.sg

Abstract. This paper introduces a possible approach for evaluating and predicting listeners' emotional engagement during particular musical performances. A set of audio parameters (cues) is extracted from recorded audio files of two contrasting movements from Bach's Solo Violin Sonatas and Partitas and compared to listeners' responses, obtained by moving a slider while listening to music. The cues showing the highest correlations are then used for generating decision trees and a set of rules which will be useful for predicting the emotional engagement (EM) experienced by potential listeners in similar pieces. The model is tested on two different movements of the Solos showing very promising results.

1 Introduction

"Io la Musica son, ch'ai dolci accenti
So far tranquillo ogni turbato core,
Et or di nobil ira et or d'amore
Posso infiammar le più gelate menti"

Alessandro Striggio for the libretto of Claudio Monteverdi "Orfeo" (Mantua, 1607).

Whether music is actually able to induce strong emotions in listeners or not is still a topic on which the scientific community frequently debates without being able to find a common view, see, for example, [1]. Nonetheless the research on this topic is very lively all around the world and studies that tried to correlate several aspects of musical structures to emotional reactions have produced fundamental works, from the pioneering [2] to [3] and, more recently [4] , while studies focusing on the analysis of musical performance aspects, such as tempo and articulation, are summarized in [5].

Lately, there have been also noteworthy experiments aimed at correlating emotion adjectives with acoustical cues focusing on different perception levels [6] and at measuring an emotional engagement of listeners while correlating the results with video and audio data streams, like in [7], following the ideas and approach proposed in [8].

In any case, regardless of today's different points of view, there are plenty of historical documents showing that music was considered able to produce strong and sudden emotions in listeners. In the past centuries nobody would have discussed about its power to move people emotionally, as clearly shown by the introductory excerpt to this paper taken from the beginning of Monteverdi's Orfeo (1607) or by XVIII century reports of performances by singers such as Farinelli, able to temporarily heal Felipe V from his depression, or Pacchiarotti, who forced a whole orchestra to stop during an opera performance since he moved all of them to sighs and tears, see [9], or even to XIX century hysteric reports of Paganini's concerts, see [10]. So the hardest problem doesn't seem deciding whether music produces emotions or not but how to scientifically measure this effect and how to produce it at will.

This paper aims at proposing a possible model for predicting listeners' emotional engagement by using a carefully selected set of sonological cues and is divided in the following sections:

- The measuring technique
- Experiment setup and data acquisition
- Cues selection and extraction
- Choosing the best cues for predicting emotional engagement
- Using the cues for generating decision trees and rules
- Testing the rules
- Discussion / Conclusions.

2 The Measuring Technique

The measuring technique for evaluating listeners' emotional engagement is a very difficult problem to solve in its own right: probably the best solution would be to design some systems that do not require a conscious feedback from the listener (such as electrodes and sensors measuring heart beat, skin conductivity, blood pressure etc.) but since such experiments are extremely difficult to be arranged properly and to make testing people feel comfortable, the most common tool used so far in this kind of experiments is a simple slider that should be moved by the listener when he/she feels the music is inducing some emotional response on him/her.

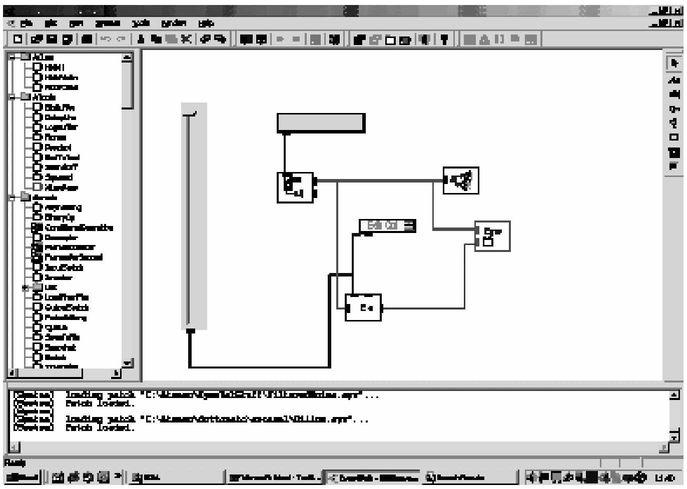
This is obviously quite risky since it would be very easy for inexperienced listeners to simply track a particular aspect of the performance, like the volume, and not any emotional effect such music is actually producing on them.

Due to this problem, we believe that, for having a meaningful experiment with this very basic tool, not only the choice of the music pieces is critical but also the people who are chosen as subjects should be very well instructed on what to do and, whenever possible, they should also have had some previous experience in these kind of experiments so as to avoid the risk of recording fake measurements as much as possible.

3 Experiment Setup and Data Acquisition

Although being aware of the “volume effect” issue just explained, we decided to use a tracking slide in a patch (figure 1) implemented within the EyesWeb open platform [11], a software especially designed for processing video and audio data streams, and then to select a small group of three people among researchers in the field who knew the risks involved in this approach and hence should have known how to avoid its pitfalls.

The used patch gives the listener control on a slider, ranging from 0 (no emotional involvement) to 127 (very high involvement) and then saves the slider position, along with a time stamp, to a text file for later analysis. The slide value is saved two times per second.



```
[illegible]   loading patch "C:\[illegible].eyw"...
[illegible]   Patch loaded.
[illegible]
[illegible]   loading patch "C:\[illegible].eyw"...
[illegible]   Patch loaded.
```

Fig. 1. An EyesWeb patch for tracking emotional response while listening to music

For our experiment we selected two contrasting movements from J.S. Bach Sonatas for solo violin (the Presto from Sonata I and the Largo from Sonata III) so as to look for aspects that go beyond the character of the piece but that, nonetheless, could be responsible for rising emotional effects in listeners.

The two movements were performed, on a priceless Guarneri del Gesù made in 1728, by Ms. Tanja BeckerBender, a young soloist winner of numerous international prizes and awards, and professionally recorded in an historical setting (a XIV century Abbey located near Genoa in Italy) so as to make her feel as inspired as in a real performing environment.

For the experiment, the three subjects had a chance to listen to the music first, if they were not already acquainted with these particular pieces, and then another listening session followed where they tracked their emotional engagement.

The results for the average response of the three subjects are shown in figures 2 and 3:

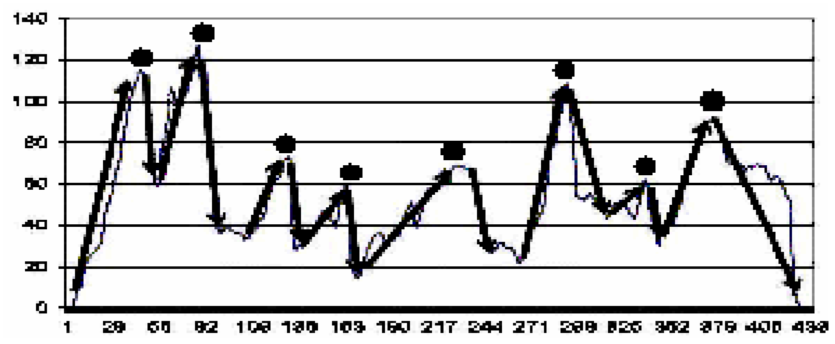


Fig. 2. Average emotional engagement response to Presto BWV1001. Y axis shows response intensity, X axis shows sample number. There are 2 samples per second. The arrows underline the rising and falling patterns. The dots point out the peaks in the response.

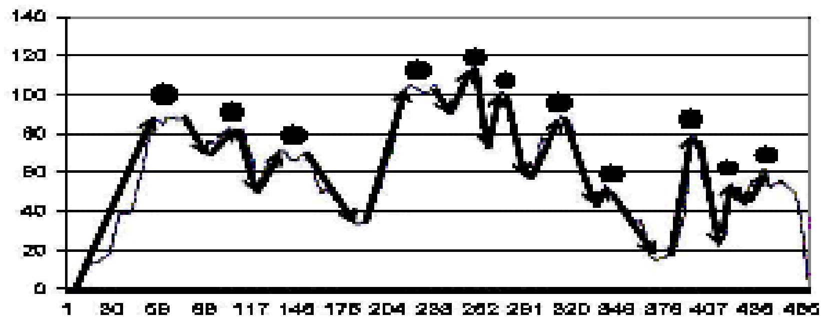


Fig. 3. Average emotional engagement response to Largo BWV1005. Y axis shows response intensity, X axis shows sample number. There are 2 samples per second. The arrows underline the rising and falling patterns. The dots point out the peaks in the response.

Interestingly, the responses of the various subjects showed peaks about the same positions and these, underlined by the dots in the figures, are well evident in the average profiles. These are the points that interest us in this study.

It is also interesting to see that most of the peaks fall down more steeply than their rises, in agreement with the idea that intense emotions do not last too long but only for a few seconds at most and then vanish quickly, as suggested in [12].

4 Experiment Setup and Data Acquisition

Our approach, for extracting data from audio recordings, starts from a very simple idea already proposed in [13] several years ago. This idea is to extract event information from a music input signal simply by rectifying and low pass filtering it, exactly like if we were working with a Pulse Amplitude Modulation (PAM) signal.

So, by squaring and filtering the signal with filters having different cut off frequencies, we get profiles bearing different information: a cut off frequency of about 20Hz will extract the envelope of the signal (we will call this note profile since it shows fast events) while reducing the cut off at around 1 Hz or less will simply take the energy of the signal (we will call this phrase profile since it shows an average behavior that changes slowly).

Then, if we compare the two profiles extracted in this way as shown in figure 4, we can divide the performance in several events that are detected when the envelope gets higher values than the energy related value.

From each of these events we can get insights on what is happening by analyzing their shape and the time that occurs in between them so as to collect valuable information regarding tempo, dynamics and articulation.

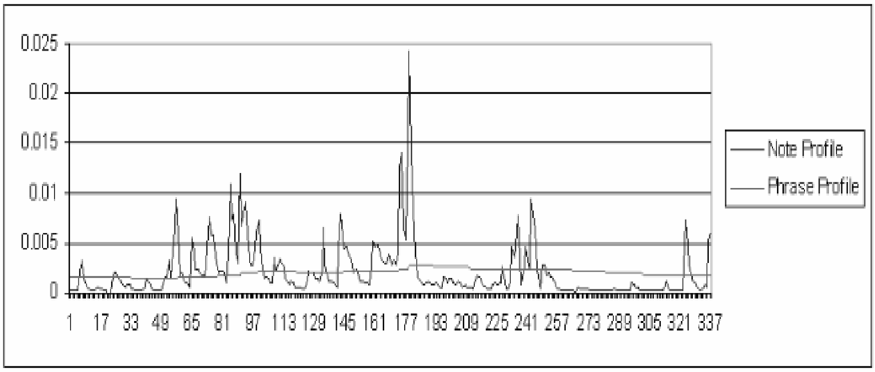


Fig. 4. Profiles obtained by squaring and low pass filtering a music file with different cut off frequencies (Y axis: amplitude, X axis: time). Note profile is the envelope of the signal and is extracted with a cut off frequency of 20 Hz, Phrase profile is the energy and is extracted with a cut off of 1 Hz.

In particular, new libraries integrated in the EyesWeb platform were developed for following this approach so as to extract a set of cues averaged across a time window where width and hop size are defined by the user at run time. For details concerning these tools see the first chapters in [14].

In this particular case the window width was set to 3 seconds for the Largo and to 2 seconds for the Presto while the hop size was 0.5 seconds, like the slider sampling rate. These values were chosen due to the experience in previous similar experiments related to playing moods and style recognition problems, like [15],[16],[17].

In particular, Eyesweb libraries were developed for extracting the following set of cues:

- **Tempo 1:** defined as the average Note Duration (DR), in seconds, of the events in the current time window.
- **Tempo 2:** defined as the number of events detected in the current time window
- **Articulation:** defined as Actual Note Duration / Inter Onset Interval averaged across the events contained in the time window.
- **Standard Deviation of Articulation**
- **Sound Level:** defined as the amplitude measured at the beginning of the event, i.e. at the intersection between the two profiles. We take the average value across the time window.
- **Sound Level Difference:** defined as the difference between the current and the previous events. We take the average value across the time window.
- **Attack Velocity:** defined as the derivative of the note profile at the intersection with the phrase profile. We take the average value across the time window.
- **Overall Energy** of the signal in the time window.
- **Energy in different frequency bands**

The latter is especially interesting since it allows us to analyze whether some bands had more relevance in the listeners' responses (the Guarneri being used by Ms.BeckerBender showed extremely strong harmonics). The available frequency range was divided in octaves so as to have a better resolution in the lower bands.

In particular the analyzed bands were:

- 172 – 334 Hz
- 344 – 689 Hz
- 689 – 1378 Hz
- 1378 – 2756 Hz
- 2756 – 5512 Hz
- 5512 – 11020 Hz

Besides these, we also looked at the pitch being played to see whether passages with rising/falling scales had any influence on the responses.

5 Choosing the Best Cues for Predicting Emotional Engagement

Now we wonder which are the factors that determined the rise and fall of the emotional engagement and whether it is possible to extract a set of rules able to explain these and then to predict them.

In particular, we are interested in analyzing what happens during the rising and falling patterns underlined in figures 2 and 3. To get an insight on this, we take the cues extracted from the performances and see how these correlate with the average profiles where the significant emotional engagement (EM) changes were identified.

The following tables (1 through 8) show the correlation coefficients R between the averaged EM and the various cues. R values with higher absolute values than 0.40 are marked in bold and R range is from –1 (variables fully inversely correlated) to 1 (variables fully correlated):

Table 1. R values for basic cues (Presto BWV1001) tracking rises in EM response

Rises (start and ending samples)	Mean DR	Notes/s	Art-Mean	ArtSD	SndLev	Sound-Diff	AttVel	Av Energy	Pitch
1.44	-0.3769	**0.4293**	-0.2360	0.1682	0.0127	-0.0959	0.1997	-0.0228	**0.4397**
54.77	**-0.7487**	**0.4186**	**-0.5953**	-0.2744	**0.8436**	0.2337	**0.6864**	0.1837	**-0.5525**
108.131	**0.4823**	**0.6648**	-0.0824	-0.3672	0.2491	0.0078	0.3304	**0.6589**	**0.5844**
140.164	0.0211	-0.0322	-0.2231	0.2996	**-0.4276**	0.0139	-0.3210	-0.2836	**-0.5498**
172.237	-0.0488	0.1461	-0.0424	-0.1689	**-0.7748**	0.0879	**-0.5161**	-0.3935	0.2865
266.292	-0.2232	0.3065	0.1222	0.0932	0.2165	0.1195	0.2266	0.2315	0.0397
317.337	-0.2179	-0.3808	-0.1258	0.1070	-0.0324	**-0.5850**	0.1460	-0.3783	0.2776
345.379	-0.3466	-0.1674	-0.0029	-0.1311	**0.6087**	-0.1907	**0.4162**	0.1624	0.2863

Table 2. R values for energy band cues (Presto BWV1001) tracking rises in EM response

Rises (start and ending samples)	172-344 Hz	344/689Hz	689/1378Hz	1378/2756Hz	2756/5512Hz	5512/11025Hz
1.44	-0.3944	0.0305	0.0351	-0.0490	-0.0016	-0.0968
54.77	0.2015	0.2224	-0.2100	-0.1428	-0.3425	-0.3685
108.131	-0.0994	0.1113	**0.6381**	**0.5963**	**0.5632**	**0.6652**
140.164	0.1159	-0.3624	-0.1735	-0.1010	-0.2429	-0.3791
172.237	-0.2988	-0.3681	-0.1392	-0.1839	0.0292	-0.0417
266.292	0.0712	0.0372	**0.4453**	-0.0008	0.2902	0.2479
345.379	-0.2927	0.0599	0.1855	0.1999	-0.1630	-0.0205
317.337	-0.2105	-0.3592	-0.1946	-0.2504	-0.1973	-0.2228

Table 3. R values for basic cues (Presto BWV1001) tracking falls in EM response

Falls (start and ending samples)	Me-anDR	Notes/s	Art-Mean	ArtSD	SndLev	Sound-Diff	AttVel	Av En-ergy	Pitch
46.54	**0.8591**	**0.8565**	**0.6496**	**0.4792**	**0.9018**	**-0.5537**	**0.9036**	**0.5629**	**0.7375**
76.92	0.2538	-0.2634	0.3679	**0.7140**	-0.3008	-0.0721	**-0.6944**	0.1448	-0.3476
131.137	**0.8838**	**0.7792**	-0.1595	**0.5149**	**-0.6044**	0.3122	**-0.4703**	**0.7001**	**0.8931**
164.172	0.1056	**0.7804**	**0.5351**	**-0.5667**	**-0.8115**	-0.3635	**-0.7359**	0.2035	-0.1393
237.251	**0.6374**	**0.8911**	-0.0302	-0.1005	-0.3339	0.0892	-0.3736	**0.6584**	**0.6254**
292.303	**-0.9148**	**0.4703**	0.2098	-0.1391	**-0.5460**	0.2005	0.1307	**-0.6154**	-0.3980
337.345	**-0.5946**	0.1086	-0.2943	**0.5403**	**0.9852**	-0.3288	**0.7372**	-0.3710	**0.4285**
379.401	-0.3763	-0.1775	-0.2744	-0.2414	0.1490	**-0.5418**	-0.0079	-0.0703	-0.0676

Table 4. R values for energy band cues (Presto BWV1001) tracking falls in EM response

Falls (start and ending samples)	172-344 Hz	344/689Hz	689/1378Hz	1378/2756Hz	2756/5512Hz	5512/11025Hz
46.54	0.0276	**0.4019**	**0.7875**	**0.5109**	**0.6028**	**0.4708**
76.92	0.1648	0.1466	0.1230	-0.0007	-0.3853	**-0.4249**
131.137	**-0.6677**	-0.0343	**0.6681**	**0.6887**	**0.7042**	**0.9151**
164.172	0.3286	0.1769	0.1783	0.3281	**-0.4137**	-0.3895
237.251	-0.3698	**-0.4454**	**0.6985**	**0.5825**	**0.6997**	**0.6599**
292.303	-0.2438	-0.3654	-0.3907	-0.2422	**-0.5723**	**-0.4649**
337.345	-0.2473	0.0095	-0.2743	**-0.8711**	**-0.7635**	**-0.7594**
379.401	-0.1480	-0.1494	0.0355	-0.0281	-0.2039	-0.1907

Table 5. R values for basic cues (Largo BWV1005) tracking rises in EM response

Rises (start and ending samples)	Me-anDR	Notes/s	Art-Mean	ArtSD	SndLev	Sound-Diff	AttVel	Av En-ergy	Pitch
1.61	**-0.4984**	0.2645	-0.2649	0.0561	**0.8676**	0.1455	**0.8334**	-0.1700	0.3324
70.108	-0.2913	-0.1254	0.0752	-0.0450	0.0690	-0.2661	-0.1120	-0.3368	-0.1107
122.141	-0.3558	0.2800	-0.3236	**0.4375**	**0.6838**	-0.1214	**0.7664**	**0.4368**	**0.4788**
190.221	-0.0666	0.3624	-0.0143	-0.3642	**0.8136**	**0.4348**	**0.5182**	**0.5438**	**0.7880**
247.261	**-0.8723**	**-0.4233**	**-0.6048**	0.0130	**0.7856**	-0.2172	**0.7713**	0.0426	**0.4659**
269.278	-0.3859	**0.6645**	-0.2753	0.1566	**-0.9628**	0.2969	**-0.8605**	**0.5298**	-0.1459
294.318	0.3120	**-0.4262**	**0.6200**	**0.4173**	-0.3906	0.2692	-0.3362	-0.1851	-0.0029
337.345	**-0.7628**	0.01264	**-0.8468**	0.1714	**0.8757**	0.1834	**0.5141**	**-0.7574**	-0.3768
377.398	-0.2318	0.2168	-0.2318	**0.4938**	**0.8095**	-0.1467	**0.6720**	0.3749	0.2187
415.423	-0.2217	**0.6668**	-0.0567	-0.1360	**-0.8021**	0.1462	**-0.4113**	-0.3382	**-0.4088**
430.442	-0.0160	**-0.6479**	**-0.5188**	0.0924	-0.0330	-0.0232	**0.6981**	-0.3376	**0.4616**

Table 6. R values for energy band cues (Largo BWV1005) tracking rises in EM response

Rises	172-344 Hz	344/689Hz	689/1378Hz	1378/2756Hz	2756/5512Hz	5512/11025Hz
1.61	-0.0961	0.1419	**0.4166**	**0.4409**	-0.2491	0.0358
70.108	-0.2408	-0.2420	-0.2951	-0.2260	-0.2535	-0.1346
122.141	0.2052	0.0821	0.3584	0.3726	**0.4364**	**0.4661**
190.221	-0.1460	0.2412	**0.6106**	**0.4667**	**0.5207**	**0.5951**
247.261	0.3018	**-0.5291**	0.0752	0.0703	0.1984	**0.4561**
269.278	-0.3877	**0.5041**	-0.1079	0.1944	**0.5331**	0.1864
294.318	**-0.4838**	-0.2312	0.1811	-0.2021	-0.1489	0.1532
337.345	-0.3199	**-0.4437**	**-0.9051**	-0.3922	**-0.7373**	**-0.8016**
377.398	-0.1284	0.3894	**0.5474**	**0.6762**	0.1750	0.1031
415.423	0.0553	-0.1245	-0.2575	**-0.4389**	-0.3558	-0.1828
430.442	**-0.5905**	**-0.6545**	-0.2484	0.1112	-0.2626	-0.1307

Table 7. R values for basic cues (Largo BWV1005) tracking falls in EM response

Falls (start and ending samples)	Me-anDR	Notes/s	Art-Mean	ArtSD	SndLev	Sound-Diff	AttVel	Av En-ergy	Pitch
76.89	**-0.5486**	**0.4329**	-0.2324	0.1130	0.2483	**-0.7637**	-0.2687	**-0.5888**	-0.2346
110.122	-0.3450	-0.1628	0.1869	-0.1208	**0.8953**	-0.0430	**0.8420**	-0.0111	**0.4130**
154.187	-0.1673	-0.3328	-0.0369	-0.0356	**0.4420**	-0.0641	0.1878	0.0287	-0.1127
234.246	0.3941	**0.7818**	0.2022	**-0.4052**	0.3087	**0.8537**	**0.9556**	**0.8234**	0.1505
260.269	**-0.6003**	**0.4051**	0.1393	-0.0524	**-0.6743**	-0.2971	**-0.8605**	0.2235	**0.6168**
278.291	0.0245	0.3117	0.2281	-0.3184	0.1652	**0.6261**	**-0.5291**	0.3099	**0.5279**
318.337	**0.4243**	**0.7934**	-0.1234	0.1351	**0.5704**	-0.1280	**0.7159**	**0.7351**	**0.7902**
344.375	0.0723	0.2748	-0.3534	-0.2941	**-0.7079**	0.2646	**-0.7723**	0.0445	**0.4542**
398.415	-0.2559	0.3618	**-0.5106**	**0.4921**	**-0.9102**	-0.2006	**-0.8918**	-0.2357	**-0.5819**
423.430	**-0.8087**	**-0.4513**	**-0.6779**	**0.6924**	-0.0530	-0.3975	**0.6109**	**-0.5915**	**0.7880**

Table 8. R values for energy band cues (Largo BWV1005) tracking falls in EM response

Falls	172-344 Hz	344/689Hz	689/1378Hz	1378/2756Hz	2756/5512Hz	5512/11025Hz
76.89	**-0.4303**	**-0.6747**	-0.3285	-0.2753	**-0.4402**	-0.1944
110.122	0.0068	-0.1936	0.1065	0.1401	0.2963	0.3914
154.187	0.1942	0.0307	0.1573	-0.0355	-0.3914	-0.2532
234.246	**0.4767**	**0.5844**	**0.7395**	**0.7854**	**0.6557**	**0.7345**
260.269	-0.2832	**-0.5671**	**0.4198**	**0.4015**	**0.4459**	**0.6038**
278.291	**-0.5535**	0.2307	0.3664	0.3400	**0.4929**	**0.4603**
318.337	0.1807	0.1757	**0.6303**	**0.5381**	**0.5272**	**0.5410**
344.375	-0.3309	-0.0249	0.0902	0.2419	0.0926	0.0519
398.415	**0.4731**	-0.0460	-0.2019	-0.2974	-0.1476	-0.1307
423.430	-0.3140	**-0.5410**	-0.2759	**-0.6739**	**-0.5926**	**-0.6685**

The significance threshold was chosen to be |0.40| since this value is high enough to show a common trend in the two series and provides a good amount of data for comparing the different cues.

By looking at these tables, we can try to identify the most relevant cues for underlining EM effects and then try to use these for generating a set of rules able to predict EM changes.

First of all, we should note that the Sound Level Cue is a very important one, as we would have expected, since it often gets high absolute values in the correlation coefficients. Nonetheless we see that we had rises marked both by positive values (volume gets louder) but also others marked by high negative ones (volume gets softer). This is true also for the falls, so the listeners have avoided the pitfalls of the volume effect problem (i.e. simply tracking volume changes).

In table 9 we see a summary of the previous tables where cues showing significantly high correlation value ($|R| > 0.40$) are written in bold.

Table 9. Number of times each cue gets a correlation value |R| > 0.40

Cue	Rises (BWV1001)	Falls (BWV1001)	Rises (BWV1005)	Falls (BWV1005)	Total
Note DR	2	5	3	4	14
Notes / s	3	5	5	5	**18**
Art. M.	1	2	4	2	9
Art SD	0	5	3	3	11
Sound Level	4	5	8	6	**23**
Sound Diff.	1	2	1	3	7
Att. Vel.	3	5	9	8	**25**
Av. Energy	1	4	4	4	13
Pitch	4	4	5	7	**20**
172-344 Hz	0	1	2	4	7
344-689 Hz	0	2	4	4	10
689-1378 Hz	2	3	4	3	12
1378-2756 Hz	1	4	4	4	13
2756-5512 Hz	1	6	4	6	**17**
5512-11025Hz	1	6	4	5	16

6 Using the Cues for Generating Decision Trees and Rules

From table 9 we see which are the most relevant cues for identifying EM changes: Notes/s, Sound Level, Attack Velocity, Pitch and the energy in the 2756-5512Hz band (from now on we will call this "Mid Harmonic Energy").

These cues data can be used for generating classification/decision trees and rules following the well known C4.5 generation and pruning algorithms proposed in [18].

The trees were built by taking cues data during the rise and fall patterns underlined previously and our aim is to classify these two categories.

The overall data set was first divided into two groups: a training set and a testing one, the latter containing 10% of the original data randomly chosen. In this way we had, for the Presto, a training set of 323 vectors (each containing the values of the five cues underlined in Table 9) and a test set of 36 vectors, while for the Largo the training set had 348 vectors and the test set 41.

A summary of the trees characteristics and testing results is shown in Tables 10 and 11:

Table 10. Classification Tree Model for Presto BWV 1001

Number of Training observations	323	Number of Predictors	5
Number of Test obervations	36	Class Variable	Rise/Fall
Total Number of Nodes	106	% misclassified	
Number of Leaf Nodes	54	On Training Data	4.33%
Number of Levels	20	On Test Data	22.22%

Table 11. Classification Tree Model for Largo BWV 1005

Number of Training observations	348	Number of Predictors	5
Number of Test obervations	41	Class Variable	Rise/Fall
Total Number of Nodes	150	% misclassified	
Number of Leaf Nodes	56	On Training Data	4.89%
Number of Levels	17	On Test Data	36.59%

Table 12. Generated Rules for EM prediction in Presto BWV1001 (R: Rise, F: Fall)

Rule	Support	Confidence
If Notes/s < 0.5 then R/F = F	1.5 %	100%
If Notes/s >= 0.5 then R/F = R	98.5%	73.8%
If Notes/s > 3.0 then R/F = R	38.3%	87.4%
If SoundLevel >= 0.00968 then R/F = F	12.1%	61.5%
If Sound Level < 0.01076 then R/F = R	91.3%	76.3%
If Sound Level >= 0.01076 then R/F = F	8.7%	60.7%
If Sound Level < 0.00827 then R/F = R	82.4%	78.2%
If Sound Level >= 0.00827 then R/F = F	17.6%	50.9%
If MidHarmonicEnergy < 0.0003889 then R/F = R	54.2%	76.0%
If MidHarmonicEnergy >= 0.0009881 then R/F = R	18.0%	72.4%
If AttackVel >= 0.01611 AND MidHarmonicEnergy < 0.0003167 then R/F = F	2.2%	100%
If AttackVel >= 0.01843 then R/F = F	7.1%	79.6%
If AttackVel >= 0.01732 then R/F = F	8.0%	65.4%
If AttackVel >= 0.00818 then R/F = R	43.3%	67.1%
If AttackVel >= 0.00599 then R/F = R	61.3%	72.2%
If Pitch >= 553 then R/F = R	43.7%	79.4%
If Pitch >= 626 then R/F = R	26.5%	77.3%

Table 13. Generated Rules for EM prediction in Largo BWV1005 (R: Rise, F: Fall)

Rule	Support	Confidence
If AttackVel < 0.00014 then R/F = F	4.0%	64.3%
If AttackVel >= 0.00712 then R/F = R	22.4%	60.3%
If AttackVel >= 0.00055 AND SoundLevel < 0.00075 then R/F = F	6.9%	95.8%
If AttackVel > 0.00356 AND Pitch >= 601 AND SoundLevel >= 0.00131 then R/F = R	11.2%	87.2%
If SoundLevel >= 0.00371 then R/F = R	42.0%	61.6%
If Pitch >= 531 then R/F = R	64.9%	61.5%
If Pitch >= 712 then R/F = R	21.3%	64.9%
If Pitch < 495 then R/F = F	25.3%	61.4%
If Pitch < 357 then R/F = F	4.3%	80.0%
If Notes/s < 0.333 then R/F = F	1.2%	100%
If Notes/s < 0.666 then R/F = F	5.2%	61.1%
If Notes/s >= 0.666 then R/F = R	94.8%	54.4%
If Notes/s >= 1.666 then R/F = R	37.6%	61.1%
If MidHarmonicsEnergy >= 0.002352 then R/F = R	16.7%	62.1%
If MidHarmonicsEnergy < 0.00001 then R/F = F	10.1%	62.9%

By analyzing the trees we can produce a set of rules to underline some common aspects found while going through the various branches.

These rules can be sometimes overlapping and redundant and, among those generated by the system, the most interesting ones, which are evaluated by looking at the support (i.e. how much of the original data they can be applied to) and confidence (i.e. how much of the data they classify correctly) percentages, are shown in Tables 12 and 13.

7 Testing the Rules

As we can see from the results showed in the previous pages, the pruned trees classify quite well most of the data but the test sets show some problems for correctly identifying the falls which were often misunderstood as rises, hence the relatively high values shown in tables 10 and 11.

Anyway, when classifying test data, the most important thing to look at is the resulting average behavior and, since the training was carried out with only two classes (rise and fall, no flat lines) it is understandable that single vectors can be misclassified.

Regarding the generated rules, it is interesting to see whether they can actually predict the emotional response of a listener in a similar piece.

To verify this, we chose the Preludio from the Partita III to test the rules generated previously and related to a fast piece (Table 12), and the first half (repeat included) of the Andante from the Sonata II for the rules relating to a slow piece (Table 13).

The rules were implemented in MatLab and weighted according to their confidence percentage (so rules with higher confidence have stronger weights) then, by feeding the cues to the MatLab file, the system evaluates whether the current vector would provoke a rise or rather a fall in the emotional response of the listener. In this way it is able to propose a possible "EM" profile that, for the Preludio, is shown in figure 5 (the graph is obtained by adding 1 when the resulting sum of the weighted rule values predict a rise, subtracting 1 when the rules predict a fall, doing nothing when there is a tie. The overall value can not go below zero).

Now we should compare this profile with that of an actual listener and see whether we have similarities.

Subject three of the previous experiment was hired again and had another session like the one explained at the beginning of this paper. His EM profile is shown in figure 6. Both profiles in figures 5 and 6 have been normalized to 1.

As we can see, the two profiles in the above figures show obvious similarities, and both plots are divided in a first introductory part featuring three main low peaks, a central high zone and an ending with three more high rising peaks.

By analyzing Ms. BeckerBender's performance, we see it starts rather softly and then it builds up with a crescendo following the rising pitch progressions in the fast passages of the score. These are the characteristics that were identified by the system and that, probably, contributed to rise the EM value of the subject who was listening to the recording.

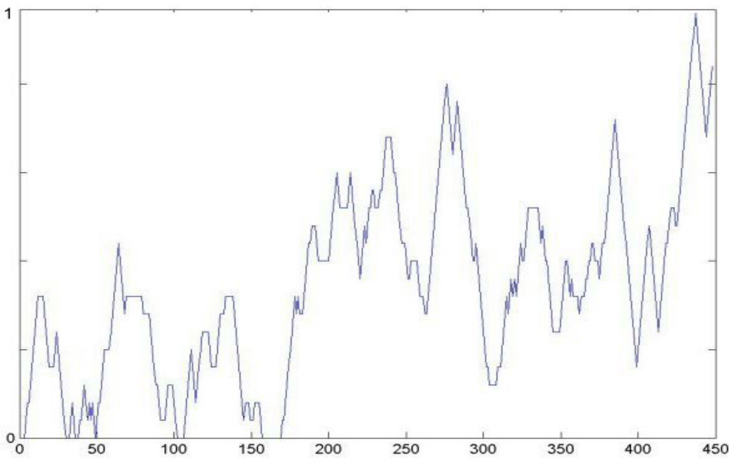


Fig. 5. EM profile predicted by the system for Preludio, Partita III BWV1006

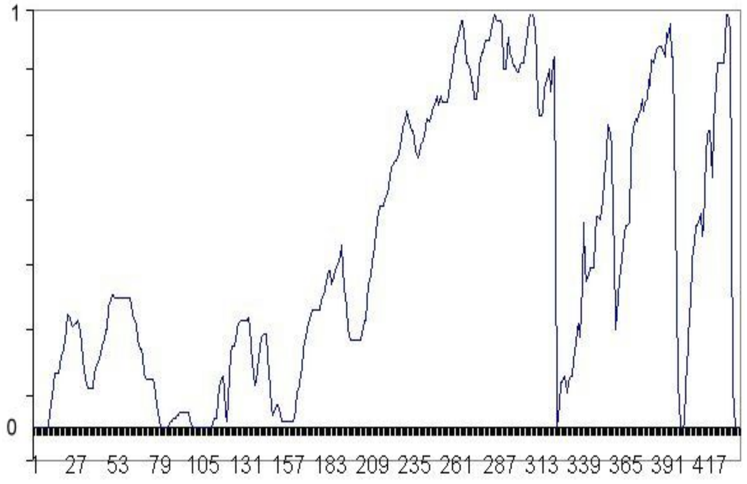


Fig. 6. EM profile as recorded by subject three for Preludio, Partita III BWV1006

Now let us test the rules generated for the slow movements. The predicted profile is shown in figure 7 while the subject response is shown in figure 8 (both have been normalized to 1).

Once more the similarities between the two profiles are striking as both of them have the EM peaks in the same positions showing that the system, driven by the previously generated rules, is able to identify the points which are most likely to produce an EM in listeners.

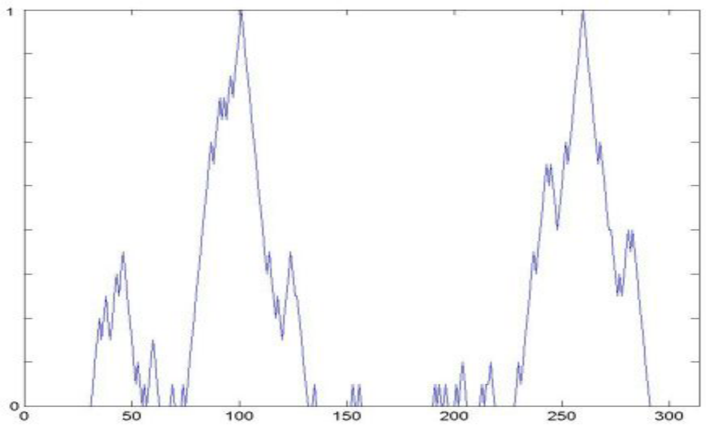


Fig. 7. EM profile predicted by the system for Andante (1st Half), Sonata II BWV1003

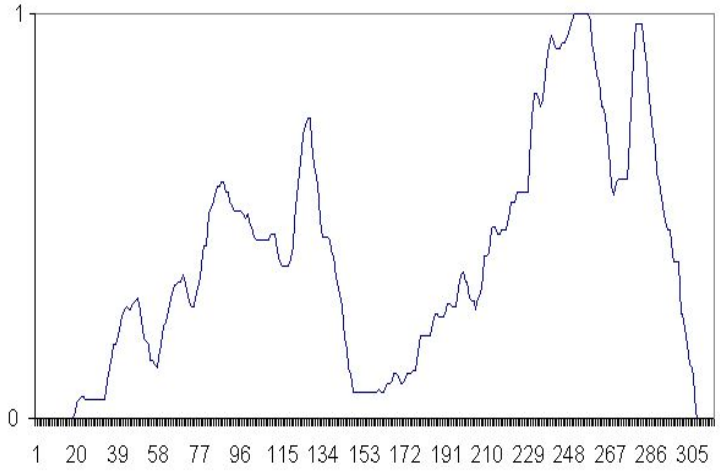


Fig. 8. EM profile as recorded by subject three for Andante (1st Half), Sonata II BWV1003

It is very interesting to compare how the system and the listener faced the repeat of the movement which was performed by the artist emphasizing different aspects such as dynamics and articulation patterns.

At the beginning of the repeat the system identified only some little spots of possible EM, but not enough to bring the emotion measures to a high level as it did during the first time. We should note that the system has no "memory" of what happened when the music was played for the first time. The listener, instead, had this knowledge and, as noted while commenting the results after the experiment, the emotional engagement felt during the first time influenced and amplified the measurements during the second listening, adding more involvement and reaching higher EM levels from the very beginning of the repeat.

8 Discussion / Conclusions

As the figures in the previous section show, the proposed model is able to predict new particular responses of one of the training subjects effectively, within this particular musical repertoire.

Despite the limited data available, this achievement looks particularly interesting since, as shown by tables 1 through 8, it seems to have successfully avoided the most common and dangerous pitfall such kind of basic experiments can easily fall into due to inexperienced subjects: the "volume effect", i.e. simply tracking loudness or any other particular cue, as explained in section 2. This aspect further validates its results since this problem was a possible major objection to other recent papers on this subject (for example, [19]) where, despite the very interesting results, such issue could have aroused during the data gathering process.

Moreover, the experiment discussed suggests several possible applications that could also be of interest for commercial products: for example an EM prediction system could be used to develop a more complex and flexible software able to predict listeners' emotional engagements in particular pieces and hence it could be used to have predictions about the possible success, i.e. sales, a particular piece/song will have among a targeted audience (a piece that shows high predicted EM and well defined peaks is more likely to be enjoyed and, hence, successful).

Such approach could also be useful in music teaching software to evaluate students' performances and even in totally different applications such as querying databases: for example selecting pieces whose emotional profile looks close to that of a given song, could be a reliable way for suggesting other possible items of interests to customers in a shopping environment such as Amazon or others.

Of course, to achieve these results, a lot of work is still needed to refine the research and make the systems more general and robust. Anyway the results obtained during the development of this research are very promising and show that the audio cues being selected are really meaningful to objectively describe and quantify several aspects of music related to emotions which, so far, could only be explained in often ambiguous and subjective words such as those used for describing particular moods and "sound colors" in music playing.

References

1. Scherer K.R.: Why music does not produce basic emotions: pleading for a new approach to measuring the emotional effect of music. Proceedings SMAC 03, Stockholm, Sweden, (2003) 25-28
2. Cooke D.: The Language of Music. Oxforf University Press (1959)
3. Sloboda J.: Music structure and emotional response: Some empirical findings. Psychology of Music, vol. 19, (1991), 110-120

4. Gabrielsson A., Lindstrom E.: The influence of musical structure on emotional expression. In Juslin P. & Sloboda J. (eds.): Music and Emotion: theory and research, Oxford University Press, London, (2001) 223-248
5. Juslin P.: Communicating emotion in music performance: a review and a theoretical framework.. In Juslin P. & Sloboda J. (eds.): Music and Emotion: theory and research, Oxford University Press, London, (2001) 309-337
6. Leman M., Vermeulen V.,De Voogt L.,Taaelman J., Moelants D., Lesaffre M.: Correlation of Gestural Musical Audio Cues and Perceived Expressive Qualities. In Camurri A. & Volpe G. (eds.) : Gesture Based Communication in Human-Computer Interaction, LNAI 2915, Springer (2004) 40-54
7. Timmers R., Marolt M., Camurri A., Volpe G.: Listeners' emotional engagement with performances of a Skriabin etude: An explorative case study. (Submitted to Psychology of Music) (2004)
8. Krumhansl K., Schenk D.L.: Can dance reflect the structural and expressive quality of music? A perceptual experiment on Balanchine's choreography of Mozart's divertimento No.15. Musica Scientiae, vol. I, (1997) 63-85
9. Barbier P.: The world of the Castrati: the history of an extraordinary operatic phenomenon. Souvenir Press (1999)
10. Berri P.: Paganini: Documenti e testimonianze. Genova (1962)
11. Camurri A., Hashimoto S., Ricchetti M., Suzuki K., Trocca R. & Volpe G.: EyesWeb – Toward gesture and affect recognition in dance/music interactive systems. Computer Music Journal, 24 (1), MIT Press (2000)
12. Picard R.: Affective Computing. MIT Press (1997)
13. Todd N.: Music and Motion: a personal view. In Proceedings 4th Workshop on Rhythm Perception, Bourges, France (1992)
14. Dillon R.: On the recognition of expressive intentions in music playing: a computational approach with experiments and applications. Ph.D Thesis, DIST University of Genoa. (2004)
15. Dillon R.: Extracting audio cues in real time to understand musical expressiveness. In Proceedings "Current research directions in computer music", MOSART Workshop, Barcelona, Spain, (2001) 41-44
16. Dillon R.: A statistical approach to expressive intention recognition in violin performances. Proceedings SMAC 03, Stockholm, Sweden, (2003) 529-532
17. Dillon R.: Classifying musical performance by statistical analysis of audio cues, Journal of New Music Research, Vol. 32 n.3, (2003) 327-332
18. Quinlan J.R.: C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, California (1993)
19. Timmers R., Camurri A., Volpe G.: On the relation between performance cues and emotional engagement, Proceedings SMAC 03, Stockholm, Sweden, (2003) 569-572

About the Determination of Key of a Musical Excerpt

Héctor Bellmann

Centre for Information Technology Innovation,
Faculty of Information Technology, QUT,
126 Margaret Street, Brisbane, QLD, Australia
h.bellmann@qut.edu.au

Abstract. Knowledge of the key of a musical passage is a pre-requisite for all the analyses that require functional labelling. In the past, people from either a musical or AI background have tended to solve the problem by means of implementing a computerized version of musical analysis. Previous attempts are discussed and then attention is focused on a non-analytical solution first reported by J.A.Gabura. A practical way to carry it out is discussed as well as its limitations in relation to examples. References are made to the MusicXML format as needed.

Keywords: key, key change, tonality, dot product, MusicXML, surface features.

1 Introduction

An essential information for most of the possible variables of interest that could be derived either from the horizontal or vertical dimensions of music is the knowledge of the key at every point in the score. Traditionally, computer music studies have been performed on the basis of a small number of examples selected or transposed to be in C major or A minor, as well as verified to be non-modulating. This is the easy way to circumvent the problem of identifying the tonic for the purpose of functional labelling. The great majority of music pieces do not comply with these requirements. Even in the early 18th century, music was likely to modulate or at least tonicize within the framework of a few measures. The alternative would be to submit the music to functional analysis.

In the music score the key is never explicitly indicated but merely implied. There are several ways to find the tonic by looking at the score, but none of them will work in all cases, and corroboration from different elements in the score is sometimes needed. The problem is a complex one because composers work in the manner of programmers who do not document their programs. Thus, a composer could establish a basic tonal plan for a work, and realize it by using the appropriate chords and modulations. But the tonal plan is not shown on the score. Once this is completed, all the scaffolding has been removed. Determining the key entails some degree of analysis, for which the analyst has to do a sort of reverse engineering on the score. He figures out the tonality on the basis of certain elements starting with the key signature and the final chord, and then checks for accidentals that reveal the minor mode, suggest modulations, secondary dominants and the like.

This should not be a problem for music from the 18th and early 19th centuries, when composers followed strictly a number of well known conventions. Although there can always be exceptions – such as Jean-Fery Rebel's Cahos –, in the baroque and classical periods, establishing clearly the tonic was the first preoccupation of the composer, and any listener with a modest musical training could sing back the tonic after hearing only a few seconds of music. The analysis of the music of this period could have been argued to be 'objective'. But as we move ahead in time, the key gets progressively more blurred. Beethoven, at the opening of his 9th symphony allowed for thirty seconds of modal haze. That was an intended effect. Matters turned more complicated in the works of Liszt and Wagner. Tonal ambiguity can be purposely created, as in Liszt's "Bagatelle sans tonalité", which means that a key cannot necessarily be objectively found even in music that could otherwise be considered tonal. If the key cannot be unambiguously inferred from the surface elements, there is a potential intrusion of subjectivity, which has to be avoided at all costs.

This means that a necessary tool in computer music studies should be a method to determine the key as a point function, including the diagnostic that no key is detectable in a passage.

2 Early Attempts

Every music student knows that music theory is full of rules. These rules give the impression to be exhaustive and complete, although music theoreticians, unlike mathematicians, have never subjected their system to such logical analysis. Music studies are also full of mechanical, tiresome and error-prone chores that require little knowledge apart from counting, such as transposing. Consequently, as soon as computers became available, many researchers thought computers would be ideal tools to take care of those tasks that can be accomplished by the mere application of rules, as well as to provide insight into the rules themselves. However, the impression of logic in music rules did not survive scrutiny. In 1968, John Rothgeb pioneered the use of computers to solve the problem of harmonizing the unfigured bass with results less than satisfactory. Years later he summarized them writing that "the computer made a significant and well-defined contribution to the study by exposing deficiencies in the theories under investigation and in suggesting further lines of enquiry" [1].

One interesting attempt among the early ones was Dorothy Gross' PhD dissertation, A Set of Computer Programs to Aid in Musical Analysis [2] in which she developed a package of computer programs to do pattern tracing, thematic analysis, grouping of sonorities and harmonic analysis. Her initial goal had been to use computers to carry out full analyses. However, like other early researchers, she found that

> "music analysis with the computer has brought to light the inadequacies of existing music theory in fully describing musical attributes because the computer reveals all too clearly the gaps and loopholes in formalized theoretical systems".

Because of this, she ended limiting her attempt to duplicate "the more routine parts of quantitative analysis". Her set of programs concentrated on horizontal and vertical surface features, and her linear and thematic analyses programs were quite successful. In relation to harmony, her program listed and counted chords according to their quality and determined the chord root. But, since no attempt was made to identify tonality and scale degrees, no functional labels were assigned. However, she found that certain chords are context-dependent and their identification could not be made in isolation. The task surpassed the capability of her pattern-matching algorithms. She explained:

> “Our one program going beyond routine operations is our harmonic analysis program, which started as a small chord-labelling option and grew into a project in simulating human thought as we realized that the definitions found in textbooks were entirely insufficient for even the analysis of a Haydn’s minuet”.

A few years later, H.J.Maxwell attempted the artificial intelligence approach to identify chords and keys. In his PhD dissertation, "An Artificial Intelligence approach to computer-implemented analysis of harmony in tonal music" [3] he recognized that "there is no clear-cut, non-intuitive method for performing harmonic analysis of tonal music". Claiming the superiority of knowledge-directed intelligent methods over brute-force algorithms, he pointed out that the ability to tell a chord from a non-chord is crucial in building a computer harmonic analysis program.

Maxwell first identified the central problem saying that "chords define the existence of tonality, but the tonality in turn defines the functions of the chords". He developed an expert system based on 55 rules centered around two main problems: "Which vertical sonorities are chords" and "What is the key", while knowing that both problems are not independent. He explained:

> "Once it is decided exactly what notes are in a chord, and what key in which to analyze the chord, finding the function label is a simple matter. But the label given may, in turn, influence what notes, which sonority, should be chosen as 'the chord'. The key is also dependent on the chords that are selected for labelling because its strength depends on the functions that can be assigned to them. This is the very crux of the problem, a symmetrical dependency – that the identity of the key depends on the chord functions, while the chords and their functions are determined by the key".

Maxwell's system proceeds through several stages, first determining consonances and levels of dissonance, and on the basis of these and their metrical placement, telling chords from non-chords. Based on the chosen chords, the tonality is assumed, and the analysis proceeds from beginning to end, “analyzing as long as possible in the currently established key, and only attempting to modulate when a certain threshold of functional weakness is exceeded".

In his dissertation, Maxwell analyzed only three pieces from the French Suites of J. S. Bach. It would have been interesting if he had continued perfecting his system, but he does not seem to have done it. In 1992 he contributed an abridged version of his dissertation to the compilation "Understanding Music with AI: perspectives on music cognition" [4] but there again he referred only to the same pieces.

Maxwell's cross-dependency is a serious drawback to the analytical approach. Music cannot be dealt with as if there was an intrinsic logic to it. Composers do not follow rules and even if rules could be thoroughly established a posteriori, it can be taken for granted that every one of them would have to admit exceptions.

In 1987, the noted chemist H. C. Longuet-Higgins published "Mental Processes" [9], a collection of articles about cognition in areas of music, language, vision and memory. In the article "On interpreting Bach" he said he wrote a parsing program "for explicating the harmonic relations between the notes of a Bach fugue subject". After discussing a number of "rules", he affirmed: "A program embodying these rules – and no others – assigns all 48 of the fugues [from The Well Tempered Klavier] to their correct keys, on the basis of the notes of the subject alone". It seemed that an exciting development was going to be unveiled. Unfortunately there was no such method. What Longuet-Higgins did is consider only the pitches of the notes of the fugue subjects, and note by note from the beginning, eliminate all the keys that do not contain that pitch. When this process leaves only one key, this is assumed to be the key of the fugue. But in one out of three of the fugue subjects, in order to 'find' the key he had to resort to the additional rule that a fugue subject either begins on the tonic or the dominant of the key. Then he needed an additional rule to be able to dispose of chromatic notes. In other words, his 'method' only works for melodies that do not modulate and begin with the tonic or the dominant of the key.

3 A Non-analytical Approach

If there were an objective non-analytical method to find the key of a chunk of music on the basis of its surface features, the immediate identification of scale degrees would greatly enhance its usefulness for analytical purposes. Since any trained person can identify by ear the tonic and the mode after hearing just seconds of standard tonal music, there must be enough information in the music to accomplish this objective. The problem is how to extract it.

James Gabura, then an undergraduate at the Toronto University, presented a paper on "Computer Analysis of Musical Style" [5] in which he made several attempts to find “an objective measure of style” considering parameters such as melodic autocorrelation, chord structure, chord duration, chord type, key and modulation, with a view to obtaining an insight into the stylistic differences between the piano sonatas of Haydn, Mozart and Beethoven. To this purpose he coded pitch and duration directly from the scores. While describing his analysis of the distribution of pitches, he made the following intriguing remark:

> "It was found that the computer could determine automatically the key in which the sample was written by comparing this distribution with arbitrarily assumed distributions for all twenty-four possible keys. By this method it was possible to determine it with perfect accuracy over the range of musical examples tested".

Later on he added:

> "Algorithms were devised which can detect key and key change (modulation) within a section or movement".

And finally, in relation to modulation:

> "...it was observed that with the parameters adequately adjusted, the computer indicated key changes decisively without oscillation between the two keys present".

Unfortunately he gave no information about the "arbitrarily assumed distributions for all twenty-four possible keys" or the algorithms devised.

In his reworking of the article for the 1970 compilation "The Computer and Music" [6], he expressed it this way:

> "For each of the excerpts coded it was possible to determine the key simply on the basis of the pitch-class distribution of the excerpt. To do this the excerpt distribution is matched against a set of *key numbers*, which define the diatonic pitch classes contained in each of the possible 24 keys".

He went on to explain the matching process, giving two alternative mathematical methods, in the first of which the key is assigned to the key index for which the dot product between the distribution of pitches and the key numbers is maximum. But again he did not explain the nature or provenance of the set of key numbers. Could he had readily solved the problem almost twenty years earlier than Maxwell's work? If there was a set of "key numbers" like he suggested, it would certainly be not 'arbitrarily assumed' but a set that in an essential way represented the tonal system.

4 The Relative Frequencies of Chords

In 1937, almost thirty years before Gabura's paper, Helen Budge, pianist and music instructor, entered Teachers College as a candidate for the PhD degree, for which she produced a thesis entitled "A Study of Chord Frequencies (Based on the Music of Representative Composers of the 18th and 19th centuries)" [7]. The study was undertaken to show the relative frequency of the chords occurring in diatonic harmony, for which the statistical information had been lacking.

Since computers were not available at the time, her study was based on hand-made harmonic analyses. She selected 24 composers from François Couperin to Edward MacDowell, and analyzed a large number of their works – mostly excerpts but including some substantial works in their entirety: Handel's Judas Maccabeus, Mozart's Symphony in G minor, Schumann's Carnaval and Mendelssohn's St.Paul. A total of 65,902 chords were hand-counted of which 11,049 were chromatic. Diatonic chords were classified and tabulated, and their relative frequencies calculated.

The results showed a number of very interesting trends and figures, for example, that the classical period shows the least variety of chords, and there is a constant increase of chromaticism with time. There are also conclusions for chord usage for individual composers – typically, Wagner shows the lowest use of the tonic chord and Verdi the highest.

As could be expected, she found that chord frequencies vary along time. However, the changes in the figures are much slighter than it could have been expected. For example, the frequency of the Tonic triad goes from 22.89% in early 18th Century to 19.69% in late 19th Century. Along the period of common practice chord frequencies do not vary so much that their relative positions could change. The table shows the overall results for chords whose frequencies are higher than 1%, in relation to the total of diatonic plus chromatic chords.

Budge’s overall chord frequencies (all inversions)

Chord	Frequency
I	34.37
V^7	14.67
V	11.25
IV	7.74
II	4.52
VI	4.18
II^7	2.00
VII	1.70
III	1.13
VI^7	0.94

What this means is that these frequencies epitomize the tonal system – tonal harmony, whether from the baroque or the post-romanticism, has a strong peak at the tonic, a secondary one at the dominant, a tertiary one at the subdominant –, and the statistics for any tonal piece should match more these figures quite closely. Consequently, they provide a method to build the set of key numbers referred to by Gabura.

Now, these figures cannot be used directly as they correspond to the frequency of chords rather than scale degrees, and it is these that are needed. It would be necessary to use the table to derive the implied frequencies for all scale degrees. The task would require some additional assumptions about the distribution of loose notes, for example that it closely resembles that of chordal notes, which is not unreasonable. In addition, the information from Budge is incomplete in the sense that she counted but did not classify the dissonant chords. However, considering that these are a set of roughly fifty chords that account for only 16.76% of the total, the effects of the imprecision of an educated guess cannot have a severe effect on the outcome.

Robert Ottman, in Advanced Harmony [8] gives a list of all the altered chords that “enjoy some degree of usage”. Eliminating repetitions, – for example the chords #i^{d7}, iii^{d7}, #vi^{d7} are enharmonic inversions of the same chord – there are 52 types of altered chords left. Assigning judiciously arbitrary frequencies below 1% to each of them, leads to the following working assumptions about pitch class frequencies.

These percentages form a suitable set that could be used as Gabura's key numbers and could be construed as a "key archetype". Notice that the frequency of the dominant, both in minor and minor, is greater than the tonic. This, of course, is the result of the fifth degree of the scale being part of both the chords of greatest frequency. As expected, the differences between major and minor are concentrated in the so called modal degrees.

Working assumptions for the frequencies of chords

Scale Degree	Major mode	minor mode
I	16.80%	18.16%
#I-flatII	0.86%	0.69%
II	12.95%	12.99%
#II-flatIII	1.41%	13.34%
III	13.49%	1.07%
IV	11.93%	11.15%
#IV-flatV	1.25%	1.38%
V	20.28%	21.07%
#V-flatVI	1.80%	7.49%
VI	8.04%	1.53%
#VI-flatVII	0.62%	6.88%
VII	10.57%	3.92%

Now it becomes possible to determine the key for every point in the score. The method considers the key numbers as the components of a vector. Because there are two alternatives, major and minor, it is necessary to take the key-numbers vector as having 24 components. The accumulated durations for the 12 pitch classes become the components of another vector. Their matching is accomplished by calculating the dot product of both in all its 24 possible relative rotations. The maximum indicates the best alignment of the music vector with the key archetype. Naturally, "perfect accuracy" cannot usually be expected. If the passage is stereotypical, the result would be clear cut. But depending on the level of tonal obscurity, the method should indicate that a key is not determinable from the passage. This has to be understood to mean that the music vector is not shaped in a tonal way. Notice that the nature of the calculation always gives a best alignment for the vector. It is still necessary to determine the degree of divergence of the vectors that disqualifies the choice of key.

5 Applying the Method

For long time, computer music studies had faced the problem of the lack of a standard format for digital music files. A good number of programs have been made commercially available to carry out different tasks such as typesetting music, playing music through a MIDI port or working as sequencers, ranging from the freeware to the very expensive. But usually each one sets its own format, and conversion from one to another is generally impossible. Even as successful a program like Finale, that could become a de facto standard, uses a format that its creators have for years refused to make public. The only widely accepted standard is MIDI, but everybody who works with music notation knows that MIDI does not know about stems, beams, measures, or the difference between G-sharp and A-flat. Fortunately, in 2001, a format intended for exchange of information and analysis, MusicXML, became available on a royalty-free basis and has already been adopted by a large number of applications, raising the hopes that it would become the long waited universal translator. MusicXML comes in two formats: Partwise, in which successive measures are contained within each part, and timewise, where successive musical parts are contained within each measure. Partwise, the default option, is used by most applications. It is simple to convert from one to the other. Notice that the internal structure of a measure in MusicXML does not vary regardless of the format.

Let us consider as a first example a brief Sonata in G minor by Domenico Scarlatti, which is only 16 measures long. In order to determine the key by the method of the maximal dot product, it is necessary to collect a large enough number of notes for the calculation to be meaningful. It is possible to calculate over the whole of the piece and extract a global result. In this case, we obtain the correct result of G minor. Notice that this is not something that should be expected. Even simple works like this modulate, and no composer ever keeps a tally on how long he stayed in the main key.

One bar at a time is a natural unit to begin with. Calculating measure by measure we obtain:

Measure by measure results for Scarlatti's Sonata

Measure 1	G minor	Measure 9	B-flat Major
Measure 2	G minor	Measure 10	G minor
Measure 3	B-flat Major	Measure 11	B-flat Major
Measure 4	B-flat Major	Measure 12	G minor
Measure 5	B-flat Major	Measure 13	G minor
Measure 6	B-flat Major	Measure 14	G minor
Measure 7	B-flat Major	Measure 15	G minor
Measure 8	E-flat Major	Measure 16	G minor

Apart from the oscillation between B-flat major and G minor in measure 10, the only anomaly is in measure 8. If we examine the notes' durations (quarter note = 8) in the content of that measure we find:

C=8;   E-flat=6;   F=6;   G=6;   A=2;   B-flat=12;

This reveals that there are too few notes and too much tonic in the measure. If we remember that the key numbers say that the note with the greatest weight is not the tonic but the dominant, it is not surprising that E-flat comes as the key. The five top dot products for the measure show additional detail:

E-flat M = 1546.8124
B-flat M = 1466.3928
b-flat m = 1456.8026
e-flat m = 1349.6004
f m = 1218.507

which says that B-flat major is trailing E-flat by only 5.2%. This result suggests the convenience of adopting Maxwell's criterion to analyze as long as possible in the currently established key. Fluctuations in the key prompt to compare the figures. In cases like this where the difference between the new found key and the previous one is very low, it is reasonable to stay with the previous key.

Another brief example, Handel's Sarabande from Suite No.16, coincidentally also in G minor, shows more clearly the limitations of applying the method to measures. Whilst the global key is again correctly identified as G minor, the measure-by-measure calculation gives:

Measure by measure results for Handel's Sarabande

Measure 1	G minor	Measure 13	B-flat Major
Measure 2	G minor	Measure 14	E-flat Major
Measure 3	B-flat Major	Measure 15	B-flat Major
Measure 4	F Major	Measure 16	B-flat Major
Measure 5	C minor	Measure 17	C Major
Measure 6	F Major	Measure 18	D Major
Measure 7	C minor	Measure 19	G minor
Measure 8	D Major	Measure 20	G minor
Measure 9	G minor	Measure 21	C minor
Measure 10	D minor	Measure 22	G minor
Measure 11	D minor	Measure 23	G minor
Measure 12	D Major	Measure 24	G minor

This is a simple piece, and certainly it is not continuously modulating. The problem is due to the measures being too brief. Measure 2 only contains the G minor triad, measure 4 only the F major triad. Nobody could possibly guess better using only the information for the current measure.

The method requires enough notes for the key to be established, just as any human listener would. As it is not possible, given the nature of music, to ensure that a certain time interval will provide them, the best solution would be to program an adjustable window to perform the calculation. In this way, the window would crawl forward adding new notes and at the same time dropping the oldest ones. The result of this process is an apparently continuous function that represents the value of the dot product of the shifting group of notes considered and the set of key numbers.

As it has been mentioned, the structure of the measure in MusicXML does not vary with the format. This is certainly an inconvenience because programming a crawling window out of the information parsed from a measure is not an easy task. By and large, the notes in a keyboard piece come all mixed up. For instance, a chord is read from the bass up but this applies only the notes for the right hand. The part of the chord that belongs in the bottom staff only appear after the top one has been dealt with, so that ultimately the notes of a particular chord will appear spread along pages of XML code. In order to perform the calculation it is necessary to effect a format conversion.

The programming of a crawling window has been carried out and applied to the Sarabande. The result is represented in Figure 1 in which the vertical lines correspond to the measures.

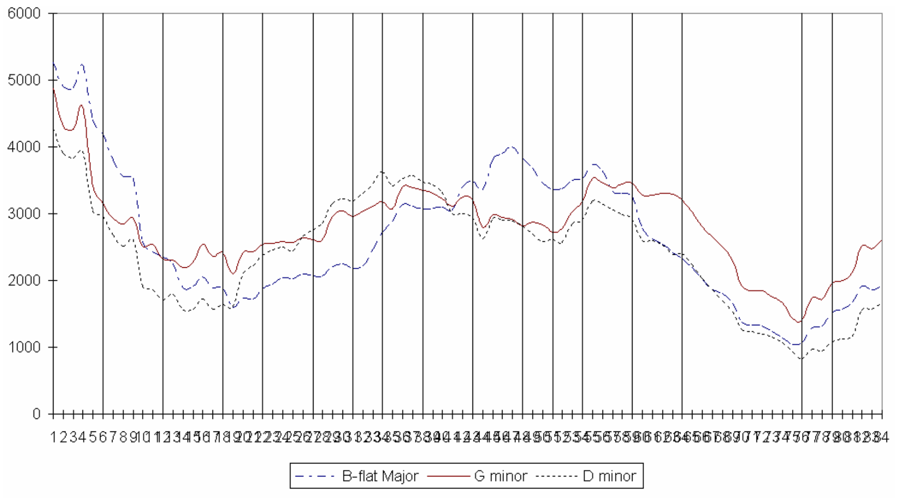


Fig. 1. Most significant keys for Handel's Sarabande from Suite No.16

The width of the window yields the first result at the beginning of measure 7, which is the one represented first on the left. It can be seen that the key is B-flat Major up to measure 9, G minor until measure 11, D minor until measure 15, B-flat Major until measure 19 and G minor again until the end. This analysis makes musical sense, but the changes appear with a delay: the first change from B-flat Major to G minor shows at the end of measure 8 whereas in the score it clearly takes place at its beginning; the change to D minor shows midway measure 11 whilst it happens at the beginning of 10; measure 13 begins in B-flat major but the program listing still has G minor at the beginning of 15. This last point is a precise reference that can be checked in the graphic. There is a triple crossing of the curves where the key changes from D minor to B-flat Major. This occurs in the score precisely at the beginning of measure 13 but the graphic shows it at three-fifths of measure 15. Finally, the last change from B-flat Major to G minor occurs at the beginning of measure 17. The graphic shows it with the same delay, one-fifth from the end of measure 19.

Music can be construed as a multi-dimension function that varies in an arbitrary way. It is difficult to observe changes in one of its variables, hidden as it is by the noise created by the variation in all the others. In order to investigate the influence of the window width, the method was tried on a fictitious composition (see appendix) where the music alternates between a tonic chord and a briefer subdominant chord for four measures (as a way to ensure the key is properly identified). It opens in C major, then at the beginning of the fifth measure switches to a distant key, C# minor, and so on. This "piece" was analyzed with a window of widths 10, 15 and 20. Figure 2 shows the results for the key of C major synchronized so that the horizontal axes coincide.

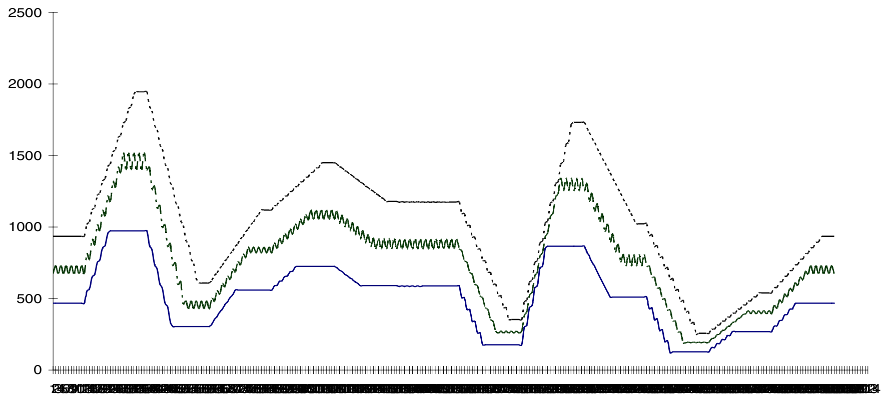


Fig. 2. Three window sizes applied to a fictitious composition

As can be observed in the graphic, the level achieved by each curve and the magnitude of the delay are both proportional to the width of the window. The transitions begin at the same time but the wider the window the longer it takes to reach their steady state. It can also be readily observed that the delay to reach steady state equals the width of the window, a result that is not immediately obvious in a real piece.

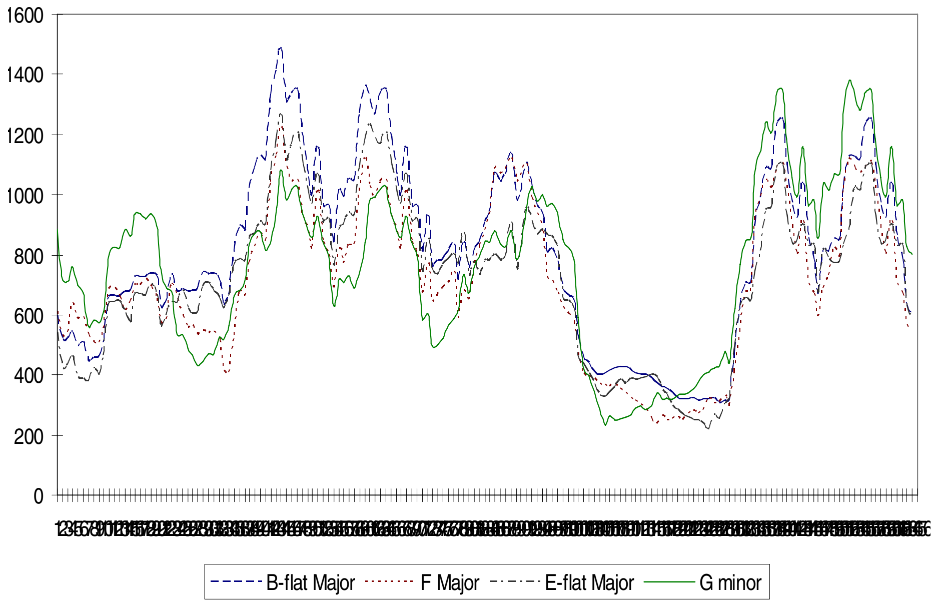


Fig. 3. Narrow window applied to Scarlatti's Sonata in G minor

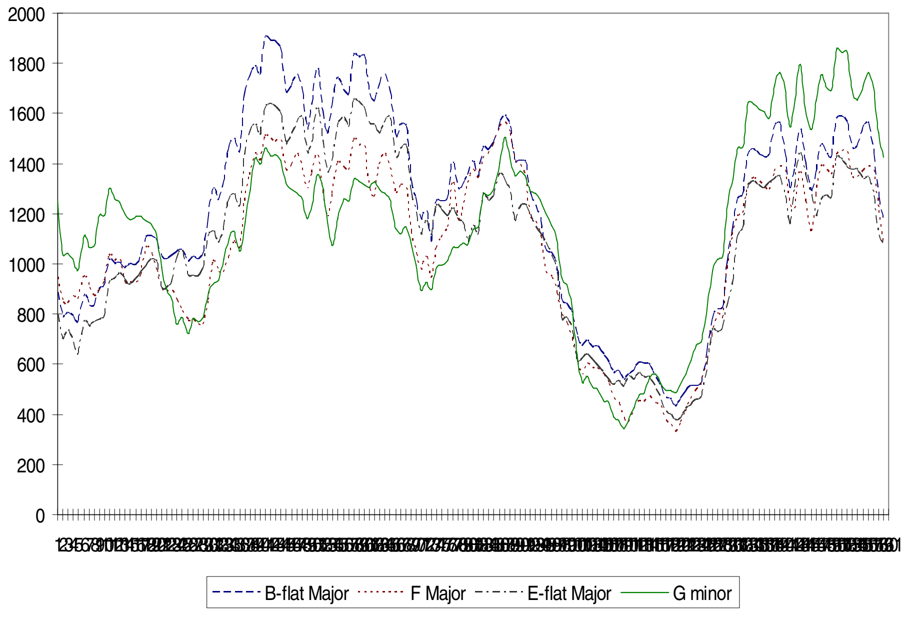


Fig. 4. Medium window applied to Scarlatti's Sonata in G minor

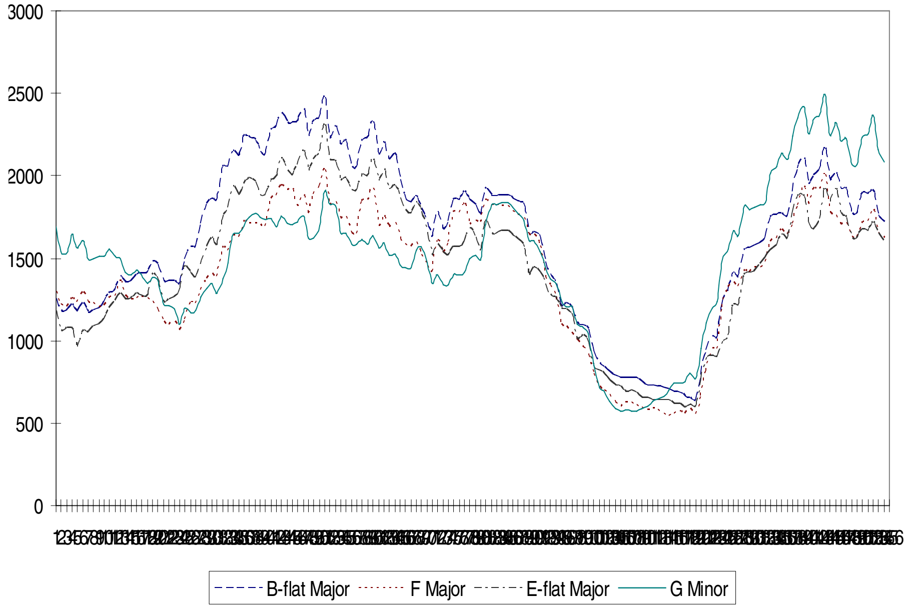


Fig. 5. Wide window applied to Scarlatti's Sonata in G minor

It is worth recalling that modulations are step functions. Determining the transition place with a window necessarily introduces an integrating effect, converting the step to a ramp.

The same process was applied to the Scarlatti Sonata, using the same three different window sizes. They are shown with windows of increasing size, in Figures 3, 4, and 5 respectively. It is easy to notice how noisy the short window is (Figure 3) which allows for a number of spurious "keys" to be identified. Notice the curves corresponding to F Major and E-flat Major, which briefly reach top value in Figure 3 whereas in Figure 4 almost match top value and in Figure 5 are close but never get to the top. The medium-size window width seems just right (Figure 4). The wide one is the most stable but it perhaps rounds things too much, so that some tonicizations could be missed (Figure 5).

In order to put the key information to practical use, the magnitude of the delay has to be taken into account. The width of the window has be large enough lest spurious transitions appear, and then the width considered to adjust the locations where they happen. Naturally, the optimum width will vary depending on the nature of the music, particularly in relation to the amount of different pitch classes present in the texture, as there must be enough for the dot product calculation to be meaningful.

6 Conclusions

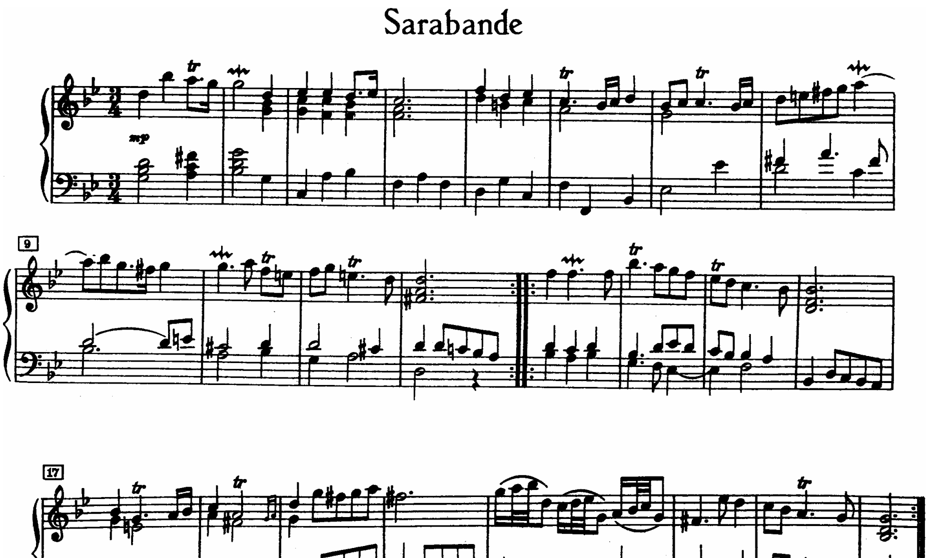
The method of assessing the key of a musical excerpt as a point function by means of aligning it with a prototypical tonal set of key numbers is consistent and effective. As it measures the extent to which a musical passage has "tonal shape", it essentially does the same job as a trained musician finding the tonic in a musical passage. Like with any other measuring instrument, what is being measured has to be known and understood for its results to make sense. Consequently, while it is possible to adopt default settings that are good for middle-of-the-road pieces, it is convenient to examine the music that has to be subject to the study in order to determine the optimal width of the measuring window. The resulting delay has to be considered to backdate the transitions. Thus, it can be adopted for musical analysis applications that require functional labelling of chords and scale degrees. It is also necessary to use the Maxwell criterion to reject modulations that are too brief when the difference between the new key and the previous one is small enough.

References

1. Rothgeb, John. Simulating Musical Skills by Digital Computer. In Schwanauer, M. S. and Levitt, D. A. (eds.) Machine Models of Music, 1993. The MIT Press, Cambridge, Mass.
2. Gross, Dorothy S. A set of Computer Programs to aid in Musical Analysis. Ph.D Dissertation, Indiana University, 1975.
3. Maxwell, H. John . An Artificial Intelligence Approach to Computer-Implemented Analysis of Harmony in Tonal Music. PhD Dissertation, Indiana University, 1984.
4. Maxwell, H. John. An Expert System for Harmonizing Analysis of Tonal Music. In Balaban, Mira, Ebcioglu, Kemal and Laske, Otto. Understanding Music with AI: perspectives on music cognition. AAAI Press/MIT Press, Cambridge, Mass., 1992.

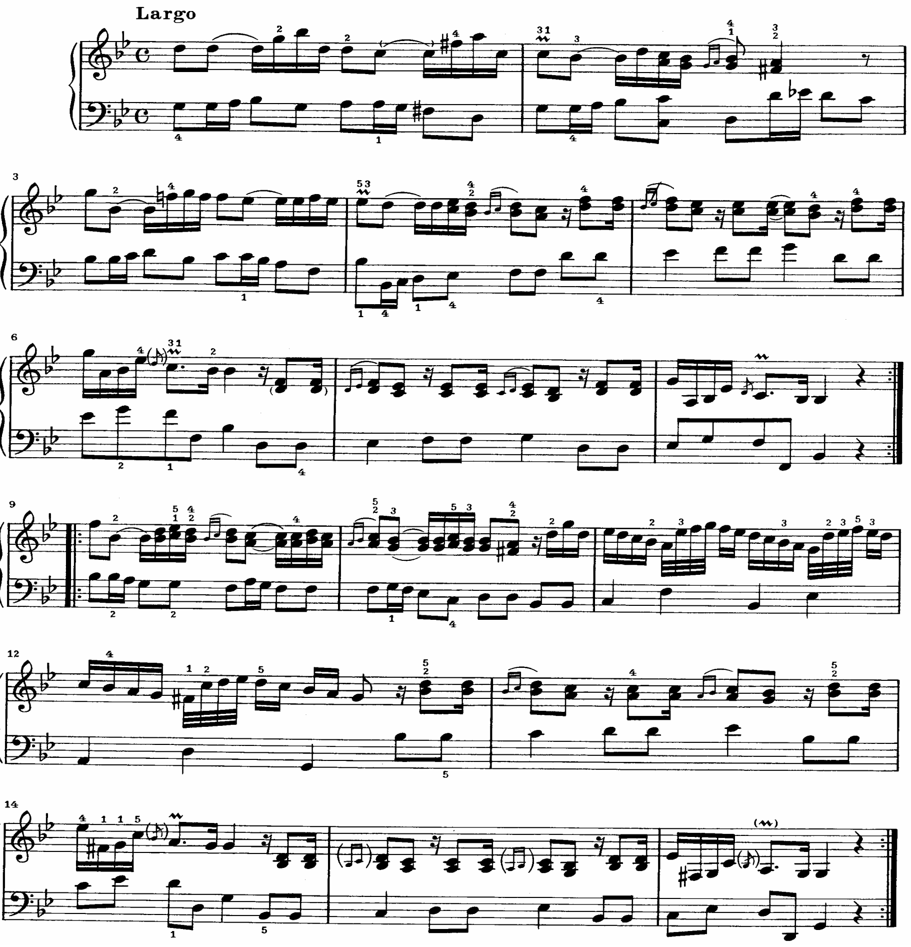
5. Gabura, James A. Computer Analysis of Musical Style. ACM Proceedings of the 20th National Conference, 1965.
6. Gabura, James A. Music Style Analysis by Computer. In Lincoln, Harry B., ed. The Computer and Music. Cornell University Press, London and Ithaca, 1970.
7. Budge, Helen. A Study of Chord Frequencies Based on the Music of Representative Composers of the Eighteenth and Nineteenth Centuries. Ph.D. Dissertation. Columbia University, 1943.
8. Ottman, Robert W. Advanced Harmony . Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1972.
9. Longuet-Higgins, H. C. Mental Processes. Studies in Cognitive Science. The MIT Press, Cambridge, Massachusetts, 1987.

Appendix

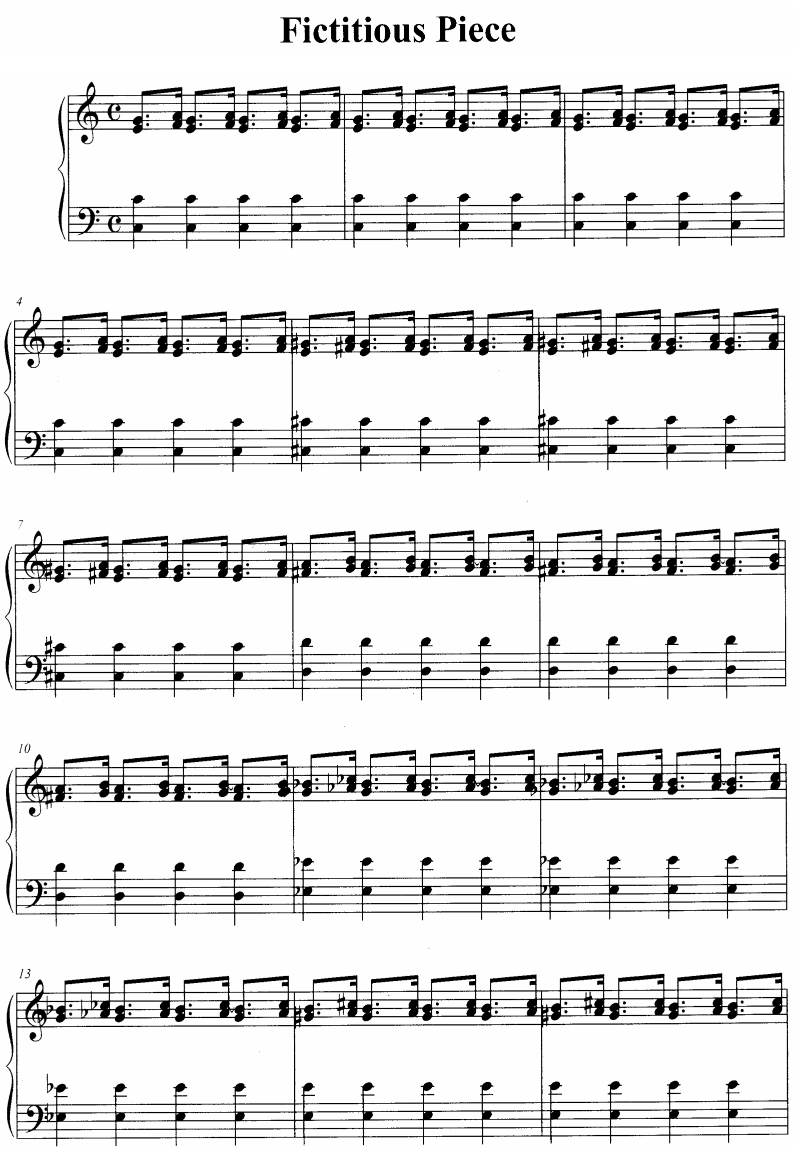


Handel's Sarabande from Suite No.16

Sonate in g



Scarlatti's Sonata in G minor



First page of a piece devised to test the effect of window size

An Interactive Musical Exhibit Based on Infrared Sensors

Graziano Bertini, Massimo Magrini, and Leonello Tarabella

ISTI-CNR, Istituto di Scienza e Tecnologie dell'Informazione A.Faedo,
Via Moruzzi 1, 56124 Pisa, Italy
massimomagrini@yahoo.it

Abstract. This paper deals with the description of the design of an exhibit for controlling real-time audio synthesis with a wireless, IR-based interface. Researching new way for playing and real-time controlling electronic music is today's hot topic in the computer music field. The goal of this specific project consists of an enjoyable, robust and reliable exhibit that gives the possibility to constantly operate with young users (especially children) and, more in general, non-expert people. Our effort has been focused to carefully design the hardware/software project, in a way that the final user will interact only with non-critical parts of the system.

1 Introduction

Interactive electro-acoustic music proposes a complete new scenario not only for the new sound palettes introduced, but also for the new kind of relationships experienced by the audience between what-is-going-on on stage and the final musical result. While a traditional music instrument is a compact tool, the new electro-acoustic instrument is a system consisting of a number of spreads out components: sensors and controllers, computer and electronic sound generators, amplifiers and loudspeakers. How to link information between the various parts of this exploded instrument is deeply correlated to new modalities of composing and performing in relationship with how the audience perceive and accept these new paradigms. While these new interaction methods for digital music and other multimedia are a hot topic in today’s research community, much of the work has remained out of reach for the general public [1], including children. The goal of our work is to use technology to support these methods of multimedia interaction for encouraging young people to more interactively explore music. The ISTI (former CNUCE and IEI) institute has deeply investigated this research field in the past years [4][5][6], so that it has been naturally involved in the realization of an educational exhibit based on a gesture music controller, shown in a local science park (Ludoteca Scientifica: http://www.ipcf.cnr.it/ludotecascientifica/).

2 Requirements and Design

This exhibit is mainly targeted for very young users, so it have to be designed following some constraints [1]:

- *Robustness of the hardware.*
- *Robustness of the gesture detection algorithm.*
- *User interface have to be designed to not only attract, but for keep hold of the child's interest.*

While the first point is rather obvious, the second ones imply that the software had to be designed to carefully and correctly respond to erratic and not logical actions, mostly ignoring them. At the same time it had to respond quickly and lively to all “logic” user actions. The last point is rather challenging, too. The whole exhibit design had to be complex enough to be fully discovered in a not too-short time but, also, its use had to be immediately understandable. As Jan Borchers suggested in [2], the typical user interface for this kind of exhibit should follow also some design goals. Basically, the GUI has to be:

innovative: exhibits must attract passing visitors and “lure” them into explore the system.

explorable: once the user has started to examine the system more closely, the interface, especially the user guidance and navigation component, has to enable her or him to actively discover successive exhibit features, offering new, motivating experiences throughout the session.

intuitive: this is a requirement for any interactive system, but with exhibits it becomes crucial: if visitors who stop in passing by to explore the system do not get along with it right away, they will walk on and leave the exhibit alone.

*non-technica*l: technical looking exhibits scare away visitors that are often computer novices. Especially an exhibit about music can provide a refreshing counterpoint in this environment if it manages to create a non-technical impression. This appearance can also stress our concern that computers should not dominate learning, but support it as creatively usable tools.

fun: an important part of the exhibit message, especially when oriented to very young visitors, is that learning could become a more enjoyable experience.

3 System Architecture

The exhibit has been built around the last generation Macintosh computers, running MacOS 10.3.2. The whole system (fig. 1) is basically split in two parts: a user area and a restricted area. The first one includes all the parts accessible to the user: computer monitor, mouse and sensing device. The restricted area is accessible only to the exhibition personnel, and cannot be manipulated by the final user.

3.1 Sensors

Designing gesture interfaces based on infrared beams has been one of our activity during the past years. The most advanced device based on this technology is the Palm Drive (former Twin Towers): a sort of three-dimensional gesture interface, built around a matrix of infrared sensors. When the Palm Drive device has been carried out

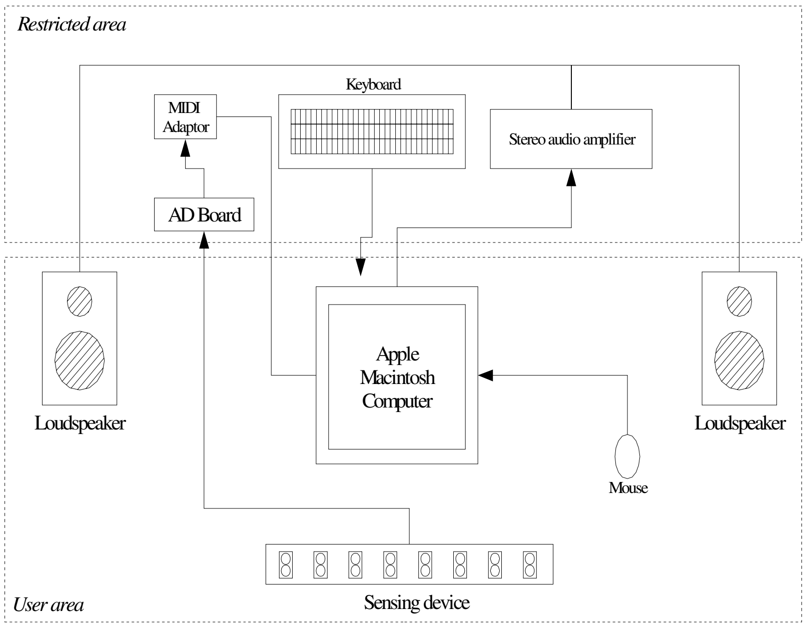


Fig. 1. The system architecture

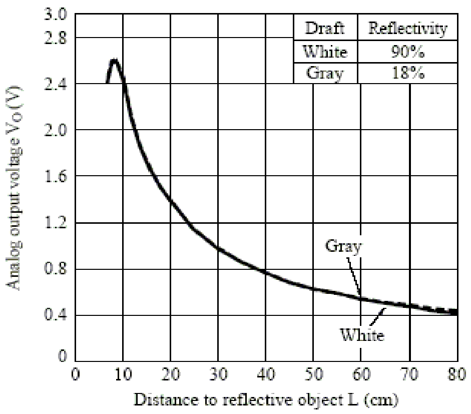


Fig. 2. Distance to volts output curve of GP2D12 infrared sensors

there weren't good integrated infrared sensors on the market, so we decided to use discrete components (infrared LED emitters and sensors). A special hardware to drive LED and managing AD conversion and multiplexing was also designed. In this present case we chose to use modern, integrated sensors instead of discrete technology: even if there is a loss in terms of design flexibility, the system development is much more rapid and less critical.

The sensor used is the SHARP GP2D12 Infrared detector. This is a small sensor, typically used in robotics. Our sensing device consists of a 1-inch thick, linear, aluminum chassis, with 8 equally spaced infrared linear sensors of this type. This sensor takes over a continuous object's distance reading and reports the distance as an analog voltage output, with a distance range of 10cm (~4") to 80cm (~30"), according to the law described in fig. 2.

3.2 MIDI Conversion Board

The role of the MIDI conversion board is converting the analog signals coming from the sensors into standard MIDI messages. MIDI is a serial HW/SW protocol commonly used for interfacing musical instruments interface.

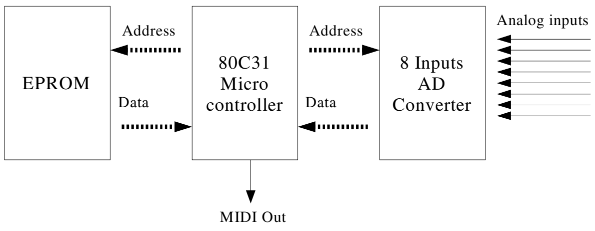


Fig. 3. MIDI Conversion board block diagram

We chose to use this standard protocol because a MIDI Management library already exists in the software development framework (see below). Instead of building ex-novo the hardware interface we decided to use a ready made one, an economic solution also in terms of efforts. The board we used is the MIDI Brain board from Paia electronics. We decided to use it because it's cheap and easy to customize, even if its design is rather old. A block diagram of the board is shown in fig. 3. The analog to digital converter used in the board is the ADC0809, which easily accepts two reference voltage for the voltage conversion span. As said before, our sensors give an output voltage ranging from 0.2 V (object at 80cm) to 2.5 V (object at 10cm), so we adjusted these reference voltage to match this range. For that we used two multi-turn potentiometers, carefully adjusted in laboratory (during the final test) to the desired voltage span. In this way all the dynamic range of the converter has been used.

4 Software

The software of the system has been written in C++ with the XCode development environment for Apple MacOS X. This is the native, free, development environment available on MacOS X. The exhibit's software has been based on the pCM++ framework, a C++ library of routines and classes (a framework) for digital audio signal processing and synthesis, developed in our lab.

4.1 The pCM++ Framework

This framework [3] is a collection of classes and functions for real time musical applications. The user can describe the synthesis algorithm creating object from a set of predefined classes, which range from standard wavetable oscillators to digital filters plucked string models, etc. A set of typical effects like delays, reverbs etc. are also included. Routing and mixing between objects is performed using simple methods calls. In a way similar to the CSound metaphor, the algorithm(s) are inserted into a function called Orchestra. The controllable synthesis parameters are then linked to a "control rate" function called Score, again following the Csound standard approach. The main difference is that the program/composition is compiled rather than interpreted, so reaching a higher degree of effective sound processing. The framework has been written in Standard C++, so that it could be easily ported to other platform. In order to ensure this portability the framework makes use of the PortAudio and PortMIDI libraries [ref], commonly used as open-source and multi platform libraries able to manage audio signals and MIDI messages in real time. They provide basic functionalities for real-time buffering of audio samples and MIDI incoming and out-going messages. In the very simple example below we show how to create two sinusoidal oscillators and how to control their frequencies with the mouse coordinates.

```
float       horiz,vert,valR,valL;
oscillator  oscL,oscR;
SimpleOrchestra  //synthesis algorithm
{
            valL = Osc(oscL,horiz*500+50);
            valR = Osc(oscR,vert *500+50);
            outLR(valL,valR); //sends signals to DAC
}


Score

{
            // set current orchestra
            orchestra = SimpleOrchestra;
            // only one section/movement.(a basic cycle)
            movement {
              //get mouse location
              mouse(&horiz,&vert);
            }
}
```

4.2 Application Architecture

The exhibit application has been designed with a *kiosk style* user interface:

- the user can interact with it only with the mouse (keyboard will be hidden)
- the user cannot switch to other applications
- the user cannot switch off the application or the OS.

Following these guidelines, the user interface has been carried using the Carbon paradigm, the procedural interface to the Mac OS X system.

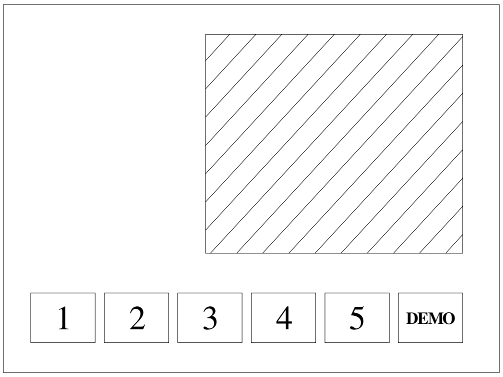


Fig. 4. The graphical user interface of the system

As shown in fig. 4 the user can interact with the application by clicking on one of the six icon-buttons of the main panel. The first five buttons allow the user to change the synthesis algorithm (presets) while by clicking on the last one, a short movie (in the area above the buttons) that shows the gesture modality of interaction for the current preset, is played.

This GUI has an obvious interaction with the lower level of the application (fig. 5), mostly based on pCM++. The audio engine, based on PortAudio, is always running, making use of the pCM++ audio synthesis support in the algorithm written in the *Orchestra* function. The MIDI data coming from the sensing device data (we use a common MIDI-USB converter for the connection) are collected by the PortMIDI library and transformed into synthesis parameters in a modality written in the *Score* function, according to the current algorithm/preset.

The two functions (Score and Orchestra) run as two concurrent processes (fig. 6), at different rates. It is important to note that the Orchestra function is called at each

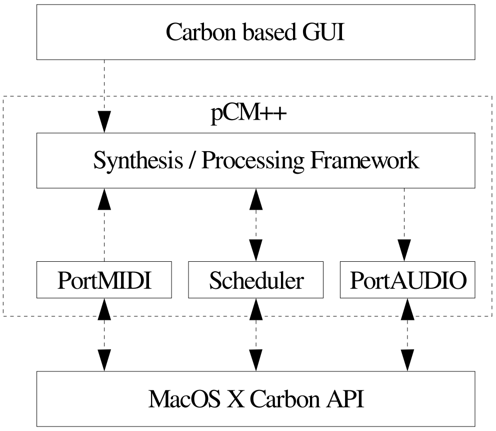


Fig. 5. The software architecture

audio buffer switch. In our case the audio buffer is 256 samples long so that, using a 44100 kHz sampling rate, the Orchestra function is called every 5.8 msec. The rate of the Score function depends on the computer performance and it is not predictable (even we can force a maximum rate): it simply has to be fast enough to ensure a small latency between the gesture and the consequent sound modification.

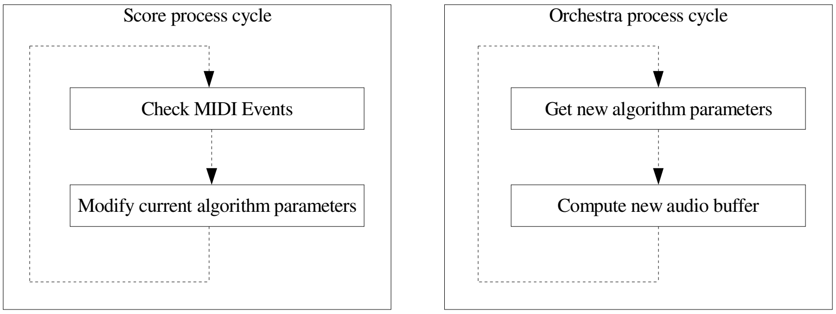


Fig. 6. Score and Orchestra concurrent processes

4.3 Interaction Models Archive and Retrievals

Even if this exhibit includes only 5 selectable audio examples, each with a different ways of interaction, we conceived it as a sort of archive in a way that it could be easily expandable in future versions of this system. Each example is described in a sort of logical record we called "interaction model", which describes both the synthesis algorithm used and the required gesture for controlling the related definition parameters. These models are then organized in a simple structure so that they could be easily and efficiently retrieved in case of a bigger archive.

The mini-archive of this exhibit includes the following 5 presets/*interactio models*:

Theremin: Simple simulation of the classic theremin (a pure sinewave), with an additional vibrato. One hand controls the amplitude while the other one can control, vertically, the frequency and horizontally the vibrato amount. A hall reverb is added as a final effect.

FM: A 3 operators cascade FM synthesizer. Left hand (more precisely, left half of the device) controls the main frequency (vertically) and switches between a set modulators frequency ratio (horizontally). The right hand controls amplitude (vertical) and modulation index (horizontal). A stereo delay is also added as an effect.

Water drops: A sample-player synth. The sample used is a water drop loop. 8 voices, with diatonic scale "pitches" are linked with each single IR sensor. The amplitude of each voice is controlled by the vertical position of the hands. A stereo delay, plus a large hall reverb are added as effects.

Harp: Based on a modified version of the classic Karplus-Strong plucked string algorithm. 8 different notes (diatonic scale) are assigned to each sensor. Hands movements trigger the harp sounds. Once the note has started, the vertical hand movement can slightly control the note amplitude and brightness. Also in this model a stereo delay and a large hall reverb are added as effects.

Storm: Similar to the water drops preset. Instead of the tuned water drops here we used a "spectacular" set of storm sounds: wind, rain and various kind of thunders are assigned to different sensors. As we experienced in previous prototypes, children love this preset.

5 Evaluation and Conclusions

A first version of this exhibit has been installed for a month in the *Ludoteca Scientifica* (a temporary science park for children) right inside the historical center of Pisa. About 6000 children visited the exhibit during the whole month.Reactions of the young users has been very good: it seems the exhibit is very fun to use. Less young users and adults continuously asked for more detailed technical information. For this reason we decided that in a future version of the exhibit we'd put a small sticker with additional technical information, explained with simple terms, beside the exhibit location. The short-video demonstration was not intensively used: in case of first-approach problems visitors preferred to ask help to the science park personnel. As far as robustness is concerned we found that the project was well made: indeed, hundreds of young users used the system 8 hours per day, and no operating faults have been encountered. We also thought to a possible variation of the system in which the "artistic" aspect is highlighted for example by adding some form of video graphics. In this case it could be also presented in modern art galleries and exhibitions.

References

1. Lee, E., Marrin Nakra, T., Borchers J.: "You're The Conductor: A Realistic Interactive Conducting", Proceedings of the 2004 conference on New interfaces (NIME04), Hamamatsu(J)
2. Borchers, J.:"WorldBeat: Designing A Baton-Based Interface for an Interactive Music Exhibit". In Proceedings of the ACM CHI'97 International Conference on Human Factors in Computing Systems (Atlanta, Georgia), pages 131-138. ACM, New York, March 1997
3. Tarabella, L.:"The pCM framework for realtime sound and music generation" in Proceedings of the XIV Colloquium on Proceedings of the XIV Colloquium on Musical Informatic (CIM2003) Firenze, Italy, May 8-9-10.
4. Tarabella, L., Magrini, M., Scapellato, G.: "Devices for interactive computer music and computer graphics performances" Procs of IEEE First Workshop on Multimedia Signal Processing, 1997.
5. Tarabella, L., Bertini., G., Boschi, G.:"Data streaming based controller for real-time computer music", Proceedings of the International Symposium on Musical Acoustics, 2001
6. Tarabella, L., Bertini, G.:"About the role of mapping in gesture controlled live computer music", CMMR 2003, LNCS 2771.

Appendix: Exhibit Photo

Metris: A Game Environment for Music Performance

Mark Havryliv and Terumi Narushima

Faculty of Creative Arts, University of Wollongong,
Wollongong NSW 2522, Australia
mhavryliv@hotmail.com
tn649@uow.edu.au

Abstract. Metris is a version of the Tetris game that uses a player's musical response to control game performance. The game is driven by two factors: traditional game design and the player's individual sense of music and sound. Metris uses tuning principles to determine relationships between pitch and the timbre of the sounds produced. These relationships are represented as bells synchronised with significant events in the game. Key elements of the game design control a musical environment based on just intonation tuning. This presents a scenario where the game design is enhanced by a user's sense of sound and music. Conventional art music is subverted by responses to simple design elements in a popular game.

1 Introduction

In Metris the person playing the game is also a musical performer. The manipulation of sound events is an integral part of the game strategy. In the design of the sounds, an arbitrary choice has been made to synthesise bell tones based on a Japanese temple bell. The sound design is an extension of work done by one of the authors [1]; game design extends interactive strategies developed by the other author in 'Medium Racing'[1] and 'The Singing Jacket' [2], [3]. The sounds used in Metris have partials that are based on the natural harmonic series. Musical scales used are also determined by the harmonic profile of the bell.

The sound trajectory provokes a musical reaction to the player's performance in the game. In this way, a player is presented with an environment for improvisation that is based on the simple rules of the game.

2 Game Design and Composition Practice

A musical composition is like a game in that the rules and parameters to control the structure of an aesthetic experience are devised prior to its realisation in performance. In a musical work, the composer specifies how these rules and parameters should be realised by a performer over time and an ideal performance is a manifestation of the composer's artistic intentions. In a game, however, it is the player who determines its trajectory.

[1] Performed at Sonic Connections, University of Wollongong, Australia (2004) http://www.uow.edu.au/crearts/SonicConnections.html

2.1 Game Design and Algorithmic Composition

Computer games are no different from other types of games in catering for a wide range of technical abilities. Metris provides an accessible environment for musical improvisation through the principles and interface of a game. The experience is intended to remain rewarding at all levels of play.

Work has been done by others to explore the audio synthesis and manipulation possibilities of using game control information. This is often realised as adaptations of existing game engines to create sound worlds through which a performer can navigate [4], [5]. Sound is controlled by both interpretation of input events and progression through the world. This process characterises the relationship between the game performance and audio creation as a one-way data stream. Other attempts at generating sound from visually represented algorithms include processes based on Cellular Automata (CA) behaviour [6], [7], [8], [9], [10], [11], [12], [13]. Like most CA visualisations, Tetris is realised as cell behaviour on a grid and as such could be considered a candidate for similar methods of sonification. The timbral and harmonic characteristics of the resultant music, however, are often limited by realisations in MIDI and an arbitrary mapping of algorithmic data to a twelve-tone equal temperament. Moreover, in both cases, mapping is mono-directional and the musical result is not based on any attempt at interpreting an individual's reaction to the relationship between audio and visual.

Metris solves these problems by controlling real time aspects of the sound through the game play itself rather than its visualisation. Tuning principles of just intonation are adapted to determine the timbre of the instrument as well as the pitch material. The player learns to recognise connections between game movements and musical outcomes. Game decisions are based on the player's reaction to both visual and audio.

2.2 Game Actions and Musical Reponses

The game responds musically in a number of ways. While the musical output is a direct result of game play, the organisation of pitch and timbre creates something larger than the memory of individual events. Musical responses to events are crafted so that the sound is affected in different ways depending on the nature of the game event: minor events affect the sound in a subtle, repeatable way (such as microtonal pitch bends when a game block is rotated); major events have a more dramatic effect on the entire sound design (such as a distinct texture modification when a row of blocks is removed).

The player has far greater control over the total sound because of the way different events are interpreted than if game actions were mapped to discrete musical responses. Game design manages individual actions within the context of the entire game; a cohesive relationship is developed between the nature of game play and the creation of sound. This is a reflection of our use of game design principles to control the player's interaction rather than interactive paradigms traditionally found in contemporary art music.

3 Real Time Bell Synthesis

In order to have control over the tuning and textural composition of a bell at any time during the game, a real time synthesis process was developed in Java. Java was chosen for its portability, but requires a more careful management of resources than a lower level language like C or C++. Real time audio techniques developed for the Pocket Gamelan project [14], [15], where J2ME code manages audio performances on mobile phones, form the core of the audio generation in Metris. For performance and control reasons, no third-party Java audio packages are used.

A bell is defined as a collection of harmonically related sine tone generators with an associated collection of amplitude envelopes. This replicates earlier attempts to create digitally synthesised bells using Csound where an instrument object encapsulates the partial frequencies and amplitude relationships of a particular bell [16]. The frequencies are taken from the five most prominent partials of the harmonic spectrum of a Japanese temple bell, namely the 2nd, 5th, 7th, 9th and 11th harmonics [17]. These harmonics are also used to generate the tuning system for the game. The envelopes must match the sampling grain of the generators to create an accurate synthesis.

Whenever a bell is struck, it is added to the audio output stream as part of a collection of any currently sounding bells. When a bell is read, each oscillator uses a current version of its frequency and amplitude envelope to determine the correct value. The amplitude envelope is stored as a set of floating point values in the same way a *linseg*[2] is constructed in Csound. Because this is applied to the sample 'on the fly' rather than as a pre-computed array, the envelope, along with the frequency of any of the partials, may be changed at any point in the duration of a bell tone. The envelope is then applied to each of the partials at the moment each bell rings.

4 Metris Music

4.1 Tuning

The same harmonics used to generate the timbre of bell sounds were used to determine pitch material for the game. Each of the seven Metris game blocks consists of four squares. In keeping with this, a seven-note just intonation scale was generated by connecting two four-note chords (tetrads) based on harmonics found in the bell timbre.

Fig. 1 shows the formation of a scale generated from harmonics 2:5:7:9. The fractions represent musical intervals in just intonation by indicating the separation between any two pitches on the natural harmonic series: the numerator is the higher pitch and the denominator is the lower pitch. For example, 5/4 represents an interval of a major third; this occurs between harmonic 5 on top and harmonic 4 below, as can be seen in Fig. 2 which shows the first 16 pitches of the natural harmonic series of C_2. Similarly, 7/4 is a harmonic seventh; 9/8 is a major whole tone, and so on. The unison is represented as a fraction where the two pitches are identical, usually 1/1.

[2] Following the conventions for *linseg* opcode defined in Csound http://www.csounds.com

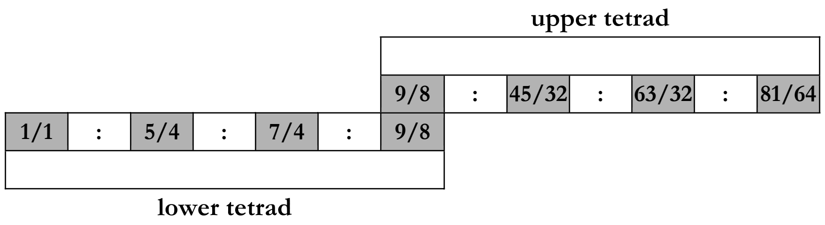


Fig. 1. Construction of a seven-note bitetradic scale generated from harmonics 2:5:7:9. The scale consists of two conjunct tetrads containing the same intervallic structure.

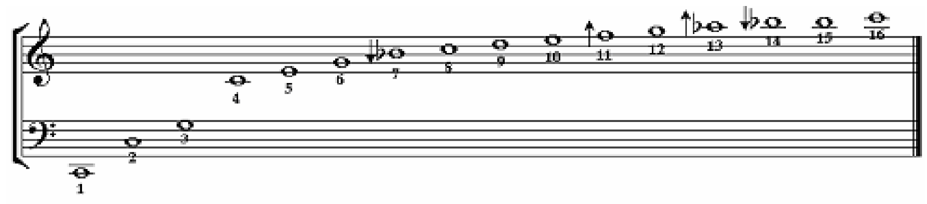


Fig. 2. Natural harmonic series (first sixteen harmonics) on C_2 [18]

The pitches of the two tetrads shown in Fig. 1 are normalised to produce the scale shown in Table 1. The ratios that appear in the second column of the table represent points on the natural harmonic series as described above. The labels that appear in the third column are the historical names given to intervals as found in many musical traditions. The linear factors that appear in the fourth column are calculated by dividing the numerator by the denominator of the ratio. These linear factors are then used to calculate the frequencies of each pitch in relation to the unison of the scale; for example, if Pitch 0 is 440Hz, Pitch 1 will be 495Hz (440Hz x 1.125).

Table 1. Intervals of bitetradic scale with generators 2:5:7:9

Pitch	Ratio	Interval above tonic	Linear factor
0	1/1	unison, perfect prime	1.000000000
1	9/8	major whole tone	1.125000000
2	5/4	major third	1.250000000
3	81/64	Pythagorean major third	1.265625000
4	45/32	diatonic tritone	1.406250000
5	7/4	harmonic seventh	1.750000000
6	63/32	octave-septimal comma	1.968750000

In Metris, each pitch of the seven-note scale thus derived is assigned to one of the game blocks as shown in Fig. 3. In this way, by using a scale generated from the same harmonics found in the overtones of the bell, the instrumental timbre is reinforced by the tuning in which it plays.

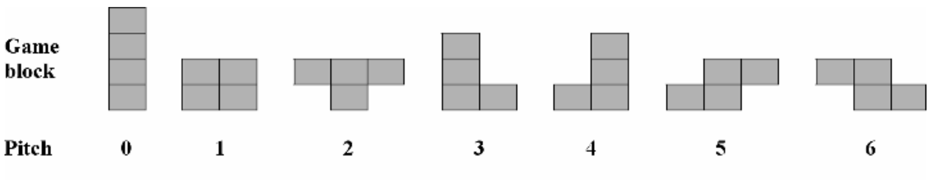


Fig. 3. Pitches 0 to 6 of bitetradic scale allocated to each of the Metris game blocks

The lower and upper tetrads in Fig. 1 are identical chords that consist of the same intervallic structure but transposed so that the highest pitch of the lower tetrad doubles as the lowest pitch of the upper tetrad. This method of constructing a scale is a variation on John Chalmers' tritriadic scales [19], [20], [21]; such a scale is labelled a *bitetradic* scale following the naming system used in Scala, a tuning software program developed by Manuel Op de Coul [22].

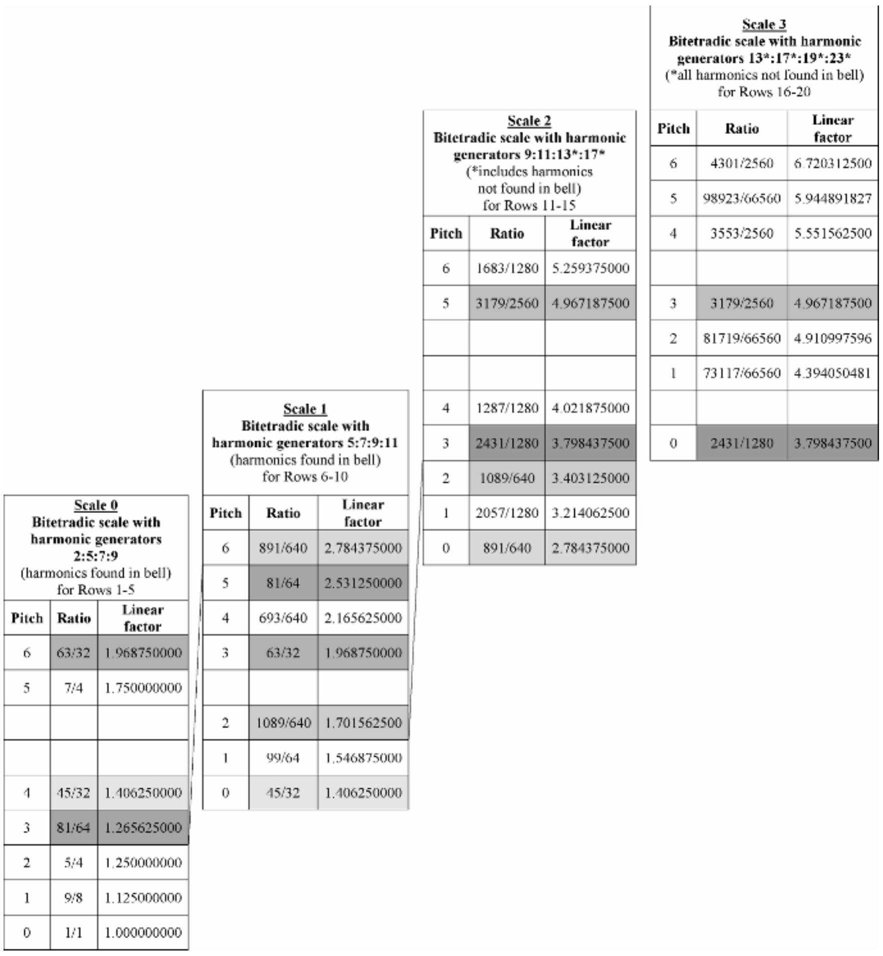
4.2 Modulation

The Metris game screen consists of 20 rows divided into four subsections of five rows each. These subsections determine regions of harmonic modulation in the game. If a block lands somewhere on the bottom five rows of the screen, notes from the original bitetradic scale with generators 2:5:7:9 (Table 1) will be played. If a block lands in a different section of the screen, pitches from another harmonic region will be played.

Modulation occurs from the original scale to other just intonation bitetradic scales built from different harmonic generators as shown in Fig. 4. Each step of the modulation introduces new intervals with higher prime-numbered harmonics[3] [23]. The original scale (bottom left of Fig. 4) and the first modulation are generated from harmonics that are present in the overtone series of the bell (harmonics 2, 5, 7, 9 and 11). The compatibility between the tuning of these scales and the timbre of the instrument reinforce each other, an idea demonstrated by William Sethares using scales derived from spectral analyses of real sounds [24]. Furthermore, the musical design of Metris includes the possibility of using scales that are not necessarily compatible with the harmonic profile of the bell. This is explored in the second and third modulations: Scale 2 includes harmonic generators that are foreign to the overtone series of the bell (harmonics 13 and 17) and Scale 3 is built from harmonic generators that are entirely unrelated to the harmonic profile of the instrument. When a game block lands in these higher harmonic areas, beating and difference tones are heard as a result of the incompatible tuning and timbre.

Pitches for all the modulations are represented in Fig. 4 as ratios and frequencies relative to the tonic of the original Scale 0. Each new scale in the series of modulations has a number of pitches in common with the previous scale; these pitches (and their octave displacements) are indicated in the diagram as shaded boxes. As the

[3] Harry Partch used the term *limits* to describe an audible characteristic of just intonation tunings that is related to specific prime-numbered harmonics. A scale belonging to a particular prime limit has a distinctive hue that makes it aurally distinguishable from scales with other limits.



Scale 0
Bitetradic scale with harmonic generators 2:5:7:9
(harmonics found in bell)
for Rows 1-5

Pitch	Ratio	Linear factor
6	63/32	1.968750000
5	7/4	1.750000000
4	45/32	1.406250000
3	81/64	1.265625000
2	5/4	1.250000000
1	9/8	1.125000000
0	1/1	1.000000000

Scale 1
Bitetradic scale with harmonic generators 5:7:9:11
(harmonics found in bell)
for Rows 6-10

Pitch	Ratio	Linear factor
6	891/640	2.784375000
5	81/64	2.531250000
4	693/640	2.165625000
3	63/32	1.968750000
2	1089/640	1.701562500
1	99/64	1.546875000
0	45/32	1.406250000

Scale 2
Bitetradic scale with harmonic generators 9:11:13*:17*
(*includes harmonics not found in bell)
for Rows 11-15

Pitch	Ratio	Linear factor
6	1683/1280	5.259375000
5	3179/2560	4.967187500
4	1287/1280	4.021875000
3	2431/1280	3.798437500
2	1089/640	3.403125000
1	2057/1280	3.214062500
0	891/640	2.784375000

Scale 3
Bitetradic scale with harmonic generators 13*:17*:19*:23*
(*all harmonics not found in bell)
for Rows 16-20

Pitch	Ratio	Linear factor
6	4301/2560	6.720312500
5	98923/66560	5.944891827
4	3553/2560	5.551562500
3	3179/2560	4.967187500
2	81719/66560	4.910997596
1	73117/66560	4.394050481
0	2431/1280	3.798437500

Fig. 4. Modulation regions in Metris: pitches from different scales are heard depending on where a game block lands. Ratios and linear factors are represented in relation to the tonic of Scale 0.

modulations move further away from the tonic, more complex ratios are involved and the intervals formed have no historical precedents. While playing the game, the user has the opportunity to move around different modulation regions depending on where they choose to land the game block.

5 Game Play and Audio Responses

Tetris has a limited number of options for controlling the performance of the game. A player can rotate the block either clockwise or anti-clockwise to place it in an effective way, and can move the block along the x-axis of the grid. The movement of the block downward is controlled by the speed of the game and is usually related to the level of difficulty, although a player can accelerate the downward progression of the block.

The game has different levels of achievement. At its most basic, the objective is to remove lines of blocks by filling an entire row. More points are awarded if more than one row is removed at a time. This introduces the possibility for daring game play and the soundtrack is designed to reward the player accordingly, as outlined below.

5.1 Rotation

When the player rotates a game block a copy of the fundamental partial of the previously struck bell is created and its frequency is adjusted +/- 3Hz. The direction of the microtonal pitch shift is determined by the direction of the rotation: A 90°clockwise rotation raises the pitch by 3Hz and a counter-clockwise rotation lowers the pitch by 3Hz.

5.2 Row Completion

When the player succeeds in completing an entire row of blocks, a sine tone texture bank (20 random sine tones) with a range proportional to the number of rows removed is played, thus destabilising any remaining tones.

5.3 Controlling the Next Block

Block creation in Tetris is usually determined by a random algorithm. Metris, however, recognises that a player may intuitively or otherwise wish to control the pitch selection in their playing. This is enabled by using the distance between the last two block placements as a percentage of the total screen size, then using that as an index to select the next block. The only exception to this is when a block lands and it overlaps with the previous block, in which case the same block is created. This was decided because the authors found repeated blocks would illuminate other aspects of the game function.

6 Performance Scenarios

The performance presentation of Metris can take many forms. One possible scenario involves Metris responding to sounds made by an instrumentalist. Metris has also been adapted for a multi-player format called Battle Metris.

6.1 Battle Metris

Battle Metris adds to Metris by allowing one player's game play to affect another player. The music production system is adapted to allow the sonic realisation of the conflict between players. The new sound represents a player's fortunes in the game and thus links the players' and audience's perception of the progression of the game.

When a player (A) removes enough rows to add them to the bottom of their opponent's (B) screen, player B's soundtrack is altered. A signal created from the amplitude-modulated sum of the partials (the *AM signal*) is added to the audio mix.

The AM signal is implemented in Battle Metris by multiplying all the partials from each bell. However, the amplitude envelopes of the partials are ignored; a partial is included in the signal multiplication at full amplitude until it is due to end. Removing the effect of the envelopes creates a significant distinction between the Japanese temple bell sounds and the AM signal.

The volume of the existing signal (the sum of the bells) is decreased to accommodate the volume of the AM signal, which is determined by the number of rows added. The partials used to create the AM signal no longer implement their amplitude envelopes; this creates a rich, *oppressive* sound. The oppressive quality refers to a sine tone with a slowly rising frequency multiplied with the AM signal, imitating the effect of the endless glissandi of Risset's implementation of a Shepard tone [25]. In the context of this game, the aural effect of the endless glissandi is similar to that of a motivator like a count-down timer in a typical digital game, in that it puts the player under pressure.

Player B can only decrease the volume of the AM signal in their mix by removing rows. The rich set of partials in the AM signal sets up beating patterns with the output of player A's channel. This is sonically exciting for the audience and player A; experience has shown it increases the level of stress in player B, as their immediate focus shifts to reducing the level of the AM signal in their mix. The oppressiveness of the AM signal complements the addition of rows as a negative effect on player B's performance.

Acknowledgments

The prototype of Metris was an extension of a Java version of Tetris by Per Cederberg [26].

References

1. Narushima, T.: Tritriadic Chimes: Bells in Just Intonation (sound installation). MicroFest 2001, Pomona College, Claremont CA (2001) http://www2.hmc.edu/~alves/microfest2001schedule.html
2. Schiemer, G., Havryliv M.: Viral Firmware: What Will I Wear Tomorrow? In: Australasian Computer Music Conference 2004. Victoria University of Wellington (2004)
3. Schiemer, G., Alves, B., Taylor, S.J., Havryliv, M.: Pocket Gamelan: Developing the Instrumentarium for an Extended Harmonic Universe. In: International Computer Music Conference 2003. Singapore University (2003)
4. delire + pix "q3apd" http://selectparks.net/archive/q3apd.htm
5. Todorovic, V. "*tadar.game music" http://www.tadar.net/
6. Brown, A., Sorenson, A.: jMusic: Music Composition in Java http://jmusic.ci.qut.edu.au
7. Chen, H.S.: Generation of Three-Dimensional Cellular Automata. In: Generative Art 2003. Politecnico di Milano University (2003)
8. Elliot, J.: Transmusic: Cellular Automaton Music (2001) http://jmge.net/camusic.htm
9. Millen, D.: An Interactive Cellular Automata Music Application in Cocoa. In: International Computer Music Conference 2004. Miami University (2004)

10. Miranda, E.R.: CAMUS: A Cellular Automata Music Generator (2002) http://website.lineone. net/~edandalex/camus.htm
11. Reiners, P.: Cellular Automata and Music: Using the Java Language for Algorithmic Music Composition (2004) http://www-106.ibm.com/developerworks/java/library/j-camusic/
12. Reiners, P.: Automatous Monk: The Cellular Automata Music Composition Program (2004) http://www.automatous-monk.com
13. Talmudi, A.K.: Evolving Decentralized Musical Instruments Using Genetic Algorithms. In: International Computer Music Conference 2004. Miami University (2004)
14. Schiemer, G., Havryliv M.: Pocket Gamelan: A Pure Data Interface for Mobile Phones. In: New Interfaces for Musical Expression 2005, British Columbia University (2005)
15. Schiemer, G., Havryliv M., Sabir, K.: Pocket Gamelan: A J2ME Environment for Just Intonation. In: International Computer Music Conference 2004. Miami University (2004)
16. Narushima, T.: Composing for Carillon: Exploring the Relationship Between Tuning and Timbre (MMus thesis) (2003)
17. Obata, J., Tesima, T.: Experimental Investigations on the Sound and Vibration of a Japanese Hanging-Bell. In: Japanese Journal of Physics 9 (1933-34) 49-73
18. Doty, D.: The Just Intonation Primer: An Introduction to the Theory and Practice of Just Intonation. 2nd ed. Just Intonation Network, San Francisco (1994) 13
19. Chalmers, J.: Tritriadic Scales with Seven Tones. In: Xenharmonikôn 9 (1986) 3-20
20. Chalmers, J.: Tritriadic Scales with Seven Tones, Part Two: Derived Forms and Structural Properties. In: Xenharmonikôn 10 (1987) 20-30
21. Chalmers, J.: Tritriadic Scales with Seven Tones, Part Three: The M->T and D->M Matrices. In: Xenharmonikôn 12 (1989) 40-68
22. Op de Coul, M.: Scala Home Page (2004) http://www.xs4all.nl/~huygensf/scala
23. Partch, H.: Genesis of A Music: An Account of a Creative Work, its Roots and its Fulfilments. 2nd ed. Da Capo Press, New York (1974)
24. Sethares, W.: Tuning, Timbre, Spectrum, Scale. Springer, London (1998)
25. Shepard, R. N.: Circularity in judgments of relative pitch. In *Journal of the Acoustical Society of America* 36 (1964) 2345-53
26. Cederberg, P.: Online Tetris Game (2005) http://www.percederberg.net/home/java/tetris/tetris.html

Strategies for the Control of Microsound Synthesis Within the "GMU" Project

Laurent Pottier

GMEM, 15, rue de Cassis, 13008 MARSEILLE FRANCE
dvlpt@gmem.org
www.gmem.org
Jean Monnet University, Department of Musicology,
42023 SAINT-ETIENNE CEDEX FRANCE
laurent.pottier@univ-st-etienne.fr
www.univ-st-etienne.fr

Abstract. Sound synthesis using granular (or micro-sonic) techniques offers very attractive possibilities for creating musical sounds. We present a state of the researches conducted at GMEM for an integrated microsound synthesis system. This system includes at once synthesis generators and synthesis control programs, both gestures and sound analysis based control.

1 Introduction

Sound synthesis for musical applications was for long time related to sound partial decomposition. But with the use of computers, spectrum based synthesis showed its limits. A simple spectrum is not enough for describing a musical sound (as noticed Tristan Murail [6]). Even for dynamical analysis with sophisticated resynthesis techniques, it appears difficult to create various sounds with additive synthesis (except in some sophisticated specials tools like those realized at CNMAT [4]).

Additive synthesis and subtractive synthesis were very often used during the 70ties but were putted in the background during the 80ties by FM synthesis. FM synthesis is a very economical and powerful techniques but not very intuitive to use. It can't be related to an analysis technique so it needs very complex studies for producing well-controlled sounds [5]. Another foreground technique in this period was the physical models. This techniques allow to create very realistic sounds but are difficult to control and limited to "pseudo physical" sound. They are not very general techniques.

That's why in the 90ties, sound synthesis was in disgrace, while sampling and sound processing were in progression in every styles of music. Main of the tools for Computer Music were plug-ins for sound processing and very few of them were concerned by sound synthesis (essentially old synthesizers imitating).

Now, it seems that one sound synthesis technique emerges : the granular synthesis. Introduced 20-30 years ago by composers like Iannis Xenakis, Curtis Roads[12] and Barry Truax, it was rarely used in real-time (except with very fast machines [15]) usually with Music N programs (like Music V or Csound) because of the calculation cost. With the last generation of computers, a lot of real-time programs offers granular synthesis techniques (Max/MSP, Pd, Supercollider). Granular synthesis expect the control of a very large quantity of data. Must of the programs mentioned below are using statistical techniques to control the synthesis.

At Gmem, we propose a very efficient granular sound generator and different ways to control the synthesis, from uniform random generation, to analysis based statistical control. The GMU project (GMEM Microsound Universe) started at GMEM in 2003[11]. Furthermore, we have built in Octave[1] and Lastwave[2] some tools for analysing the distributions of sound particles in noisy sounds. Finally, using OpenMusic[3], we have developed a new library to process the analysis data and to control sound synthesis with Csound[4] or Max/MSP[5]. Examples of the utilisation of these tools in musical situation will be presented.

2 The Sound Generator

We have developed various sound generator for Max/MSP [11] . The last of them, *bufgranul~* , is the most general. It was written by Charles Bascou (previous versions by Loic Kessous and Laurent Pottier, Windows XP version by Leopold Frey). This object must be used with three arguments : the **sound buffer,** where the sound grains are picked, the **envelope buffer** that describes the grain envelope and the **number of outputs** for sound spatialization.

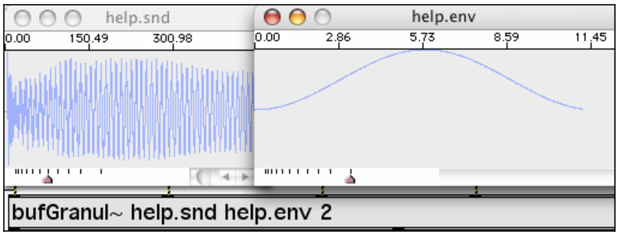


Fig. 1. The “bufgranul~” object and the buffers associated

The generator has been optimized because granular synthesis required often high polyphonies. On a PowerBook G4 867 MHz, it uses 30% of CPU with 100 grains and 75% with 300 grains.

2.1 Parameters

The inputs for the *bufgranul~* object are the following: the **triggering** of grains realized with a zero-crossing signal (can be periodic or not); the reading **position** inside the sound sample (begin parameter); the **detune** parameter, the **amplitude** of the grain, its **length**, its **stereophonic position**, the **distance position** of the source (according to Holophon spatializer algorithm [8] and the number of the buffer where the sound is stored (each grain can come from different buffers).

[1] Octave: GNU software by J. W. Eaton.
[2] Lastwave: software for signal processing by Emmanuel Bacry.
[3] OpenMusic (OM) is the Ircam visual programming environment for creating computer aided composition applications.
[4] Csound: software synthesis program (MIT) by B. Vercoe.
[5] Cycling74/Ircam.

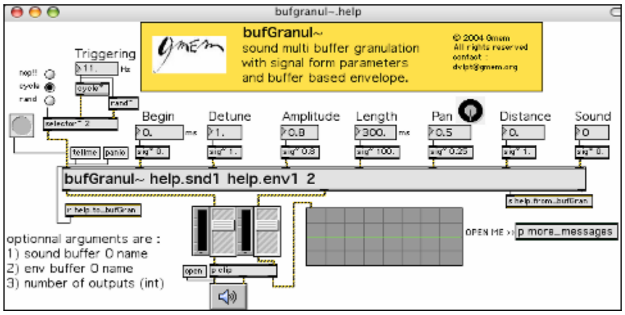


Fig. 2. The “bufgranul~” help patch

Sounds produced by the help patch are sounding mostly “electronic”. The idea of “notes” disappears replaced by a continuous flow of sound. For producing lively or natural sounds, all parameters must be modified all the time. So we need continuous controllers or automatic rules that produce flows of data.

2.2 The VNS Gesture Control

An example of a very effective way of changing parameters all the time is to use a video camera placed in front of a performer. We are often using the VNS[6] software inside Max/MSP [10]. Each small movement of the performer in different regions of the camera field can be traduced in values for the control of the synthesis. VNS is a very fast software (as fast as video can be) and very precise (a resolution of 20 bits per region!)

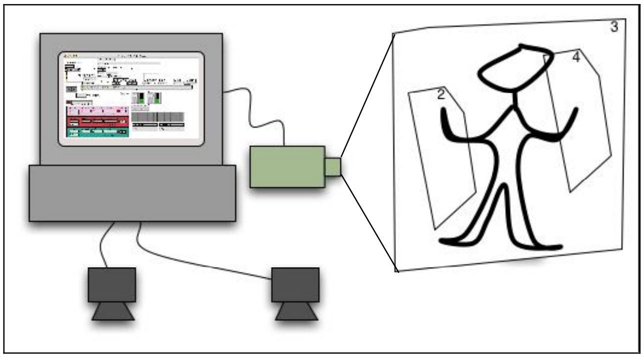


Fig. 3. The “bufgranul~” generator controlled by motion and presence of a performer using VNS software

The patch below was realized with the French composer Andre Serre-Milan for its interactive installation “Public Space 1” in December 2004.[7]

[6] Video processing library for Max/MSP by David Rokeby.
[7] Commande de ART ZOYD, coproduction ART ZOYD-Maubeuge / GMEM-Marseille.

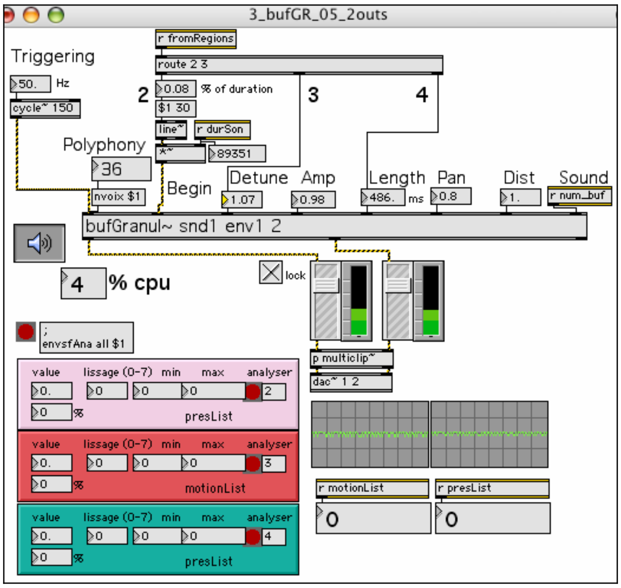


Fig. 4. The “bufgranul~” generator controlled by motion and presence of a performer using VNS software

The presence of the arm of the performer in region number 2 controls the reading position in the sound sample. In region number 4, it controls the length of the grains. Motion in region 3 controls the detune parameter (pitch).

2.3 The VNS Graphic Control

Some researches have been introduced for the control of sound synthesis parameters with pictures and video films.

In the first case, a slider controls the horizontal position of a vertical line that moves on a tree picture. This line is divided into 128 segments, in which the variation of light intensity is measured. Each value higher than zero triggers a grain. The amplitude of the grain depends on the light variation in the segment. If a group of

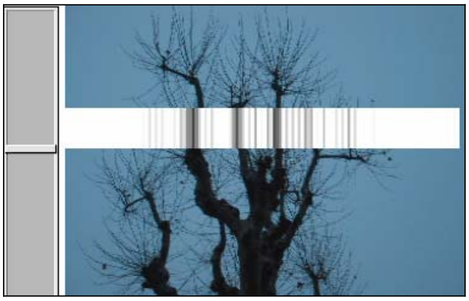


Fig. 5. A tree picture scanned for the control of sound synthesis

adjoining segments is excited, the pitch is going down (detune) and the amplitudes of the segments of the group are added. The panoramic parameter is directly set with the position (left <--> right) of the segment. So, little boughs produce high small sounds and the trunk produce a loud low sound.

Another test was done with a non-figurative video film. In this film, white geometric figures were twinkling. They were more and more numerous and smaller and smaller. With the same method as previously described, we just played the film and for each geometrical figure, a grain was triggered. The resulting sound was a little polyphony at the beginning and was like a complex noise at the end.

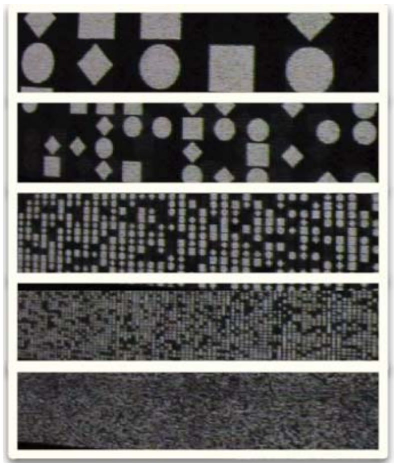
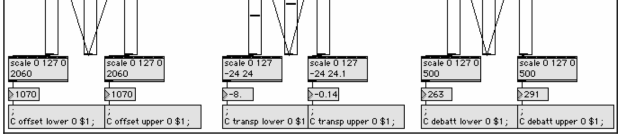


Fig. 6. Five moments in the video film that control the sound synthesis (image by A. Liberovici, in his piece "Integral")

3 Probabilities Graphical Control

3.1 Ircam GIST and Chant

The GMU project was influenced by first experiences we have done at Ircam in 1995 with the Gist software [13] on the Next ISPW music workstation. In the Gist



```
GIST CONTROL
scale 0 127 0 2060   scale 0 127 0 2060   scale 0 127 -24 24   scale 0 127 -24 24.1   scale 0 127 0 500   scale 0 127 0 500
1070                 1070                 -8.                  -0.14                  263                 291
; C offset lower 0 $1 ;   ; C offset upper 0 $1 ;   ; C transp lower 0 $1   ; C transp upper 0 $1 ;   ; C debatt lower 0 $1 ;   ; C debatt upper 0 $1 ;
```

Fig. 7. Statistical control of 3 parameters (offset = begin time, transp = detune, debatt = attack time) on the fog~ synthesizer. This patch was created for an installation of C. Ikam and J. B. Barrière "Le Messager" [8].

software, Gerhardt Eckel wrote a *fog~* object, extension of the *fof~* generator used specially for singing voice synthesis.

The *fof~* and *fog~* objects still exist on Max/MSP. Francisco Iovino and Richard Dudas wrote famous patches for the control of the parameters of the synthesis so all the rules discovered by Xavier Rodet [14] are now available in Max/MSP.

3.2 Probabilities Uniform Distribution

As on the Ircam ISPW, the first way to control automatically a flow of varying synthesis parameters that we implemented, was the statistical uniform distribution. This was made with the participation of Jean-François Oliver.

In this program, the range of the values has to be set for each parameter (min and max values) and a random generator chooses values in this area. The triggering of the

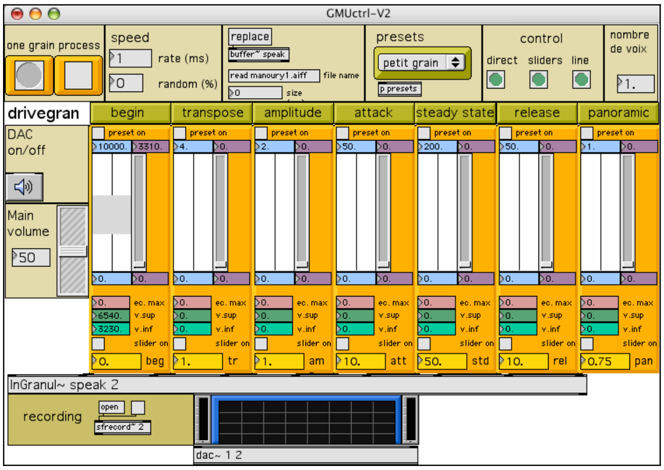


Fig. 8. Max patch for a stochastic control of the synthesis (uniform distribution)

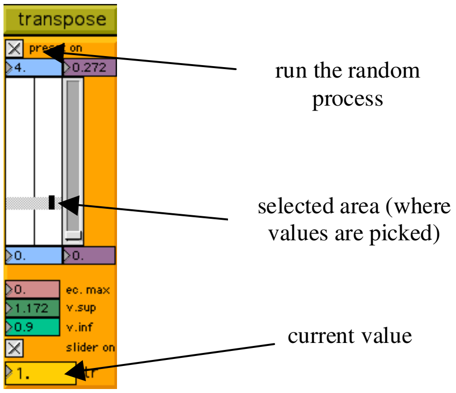


Fig. 9. A single block for the control of one parameter

grains can be periodic (with a given frequency) or not (a random value is added to the frequency).

3.3 Graphical Distributions

Uniform distributions are sounding always something like “grey”. So we decided to ameliorate the statistical control of the parameters. Charles Bascou has developed a new graphical interface for settings the probabilities of a group of values.

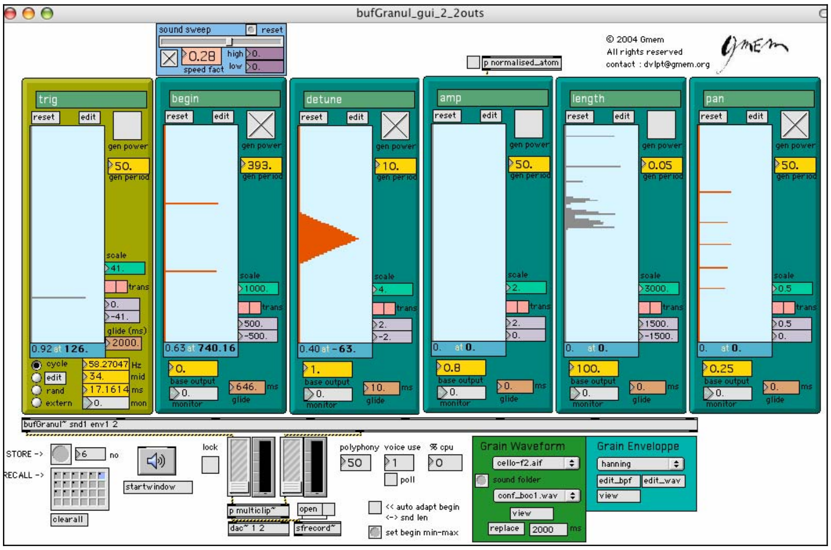


Fig. 10. Max Patch for a stochastic control of synthesis (with probability curves)

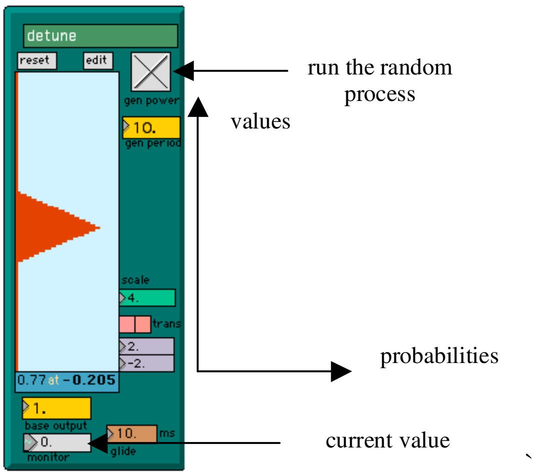


Fig. 11. A single block for the control of one parameter

With this program, the user can choose different curves for each parameter. Curves can contain only few values (like for the “begin time” or “pan” parameters in figure 9) or a continuous curve of values (like in figure 10 for the “detune” parameter).

The triggering of the parameters is done with a signal wave each time that the signal is crossing zero. If the signal is a simple oscillator with a constant frequency, the triggering is periodic.

In the next chapter, we will describe analysis tools. The data given by the analysis can be used for settings the probability curves (like for “length” parameter in figure 9).

3.4 Discussion

The control of sound synthesis in a stochastic way is very powerful. You can combine stochastic control of some parameters and deterministic control of other parameters.

For a recent work, we were using GMU for real time delaying and transposing sound produced by acoustic instruments[8]. We recorded the sound in a buffer (in loop mode) and then triggered grain at a “begin time” minus the desired delay. The triggering period was set to 25 ms and the length of the grain to 75 ms (3 times the period). The detune parameter was set to the desired pitches. Other parameters were constant or statistically distributed.

Microsound synthesis is a very efficient technique for the transposition of sounds without varying time.

The patches we have presented here, allow producing a great variety of sounds. They can be used as a polyphonic sampler (if the length parameter is equal to the length of the sampled sound) and they can be used to produce clouds of sounds, when the polyphony is high and the sizes of the grains small.

Microsound synthesis allows you to explore the transition between rhythm and frequency (triggering the grains near 10 to 50 Hz), between crunchy grains (envelope sizes less than 50 ms) and smooth grains and between spatially localised grains (fusion) and dispersed grains[9]. All these explorations can produce very strange and interesting effects on the perception.

4 Analysis-Synthesis

The exploration of sound synthesis is endless. The problem with this type of synthesis, for composers, is the lack of references. That is why we wanted to study natural sounds to establish rules for the control of the GMU synthesizer.

---

[8] You can record a sound in a buffer~ (memory space in RAM) and play with buffgranul~ the wave stored in this buffer. Obviously, there can be a delay between writing and reading, especially if you want to read faster than you write (“detune” value higher than 1). The buffgranul~ object can loop in a buffer. That means that if you begin to read a grain at the end of a buffer, it jumps to the beginning of the same buffer. It allows reading and writing with loops.

[9] The grains produced by the bufgranul~ object can be sent each one on different outputs (up to 16 outputs).

4.1 Iana Analysis Synthesis

Few years ago, we have experimented a Terhardt algorithm that can be used to extract the most pertinent partials from musical sounds [7]. This algorithm were programmed (and named "*Iana*") by Gerard Assayag and further ported on the Ircam software Audiosculpt.

In 1993, with the help of Xavier Chabot, we implemented at Ircam the "spdata" library for the PatchWork[10] environment. This library was done for reading and processing spectral data, for sound synthesis.

With this library, we could read Iana data, process them and do the synthesis with small grains (the size of the step of the analysis). It was used by the American composer Joshua Fineberg for his piece "Paradigms" (1994).

Since the Iana algorithm was ported on Max/MSP by Todor Todoroff, we have implemented an interface between *Iana* and *bufgranul~* to realize in real-time the sound analysis and the sound synthesis.

4.2 MP Analysis Synthesis

We wanted to find another technique for analysing sounds, allowing analysing noisy sounds (sounds without well defined partials) and representing signals with simple sonic entities localized in time and frequency.

We have look at the Matching Pursuit algorithm [3] that we have extended toward a sound decomposition on a set of arbitrary microsounds [2].

The idea of Matching Pursuit is to use a dictionary of functions to represent in a compact way a wide range of signal time-frequency behaviours. The traditionally used dictionary is a set of symmetric Gabor atoms indexed by their frequency and duration.

It leads us to a granular decomposition method onto a set of arbitrary microsounds. Each microsound is described by a list of five parameters: time, duration, amplitude, frequency and phase.

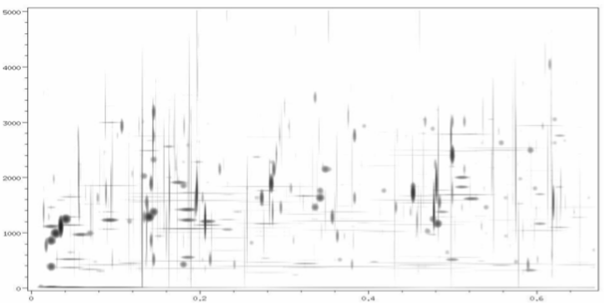


Fig. 12. The representation of a water sound with the Matching Pursuit algorithm

For some types of sounds (like water sounds), we can obtain a good description of the sound with 500 to 1000 grains per second.

[10] PatchWork (PW) was, before Open-Music, the main Ircam program for computer aided composition.

We have then realized a new library for Open-Music during a residence of the composer Tristan Murail at Gmem (2004-2005). It contains functions for reading MP analysis, writing Csound scores and process large quantity of data. A lot of programs have been written for modifying the data before doing the synthesis.

4.2.1 Proliferation

The first problem was to extend the duration of the analysed sound. Two algorithms have been proposed.

In the first algorithm we are doing random access of grains in the data file. The global density (number of grains per second) is preserved. This algorithm is simple but we lose the time progression of the grains distribution in the original sound.

The second algorithm begins with a segmentation of the data files during time. We then obtain a curve that gives the grains density d = f(t).

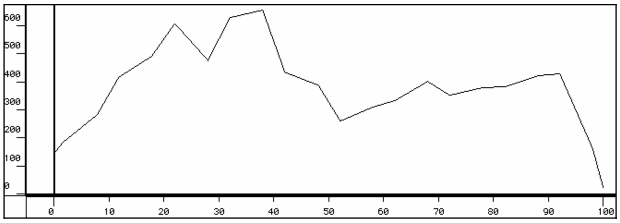


Fig. 13. Grain density curve of the analysed sound (water sound)

In a second step, we are doing random access of grains in a portion of the data file that progress in a way proportional with the desired duration of the synthesized sound. The densities are varying according to the density curve.

With this method, we can stretch a sound with great factors without problems (like x100 or x1000).

This method can be extended. We can use a curve to indicate how we want to walk inside the data file during time. For example, if we have a sea wave sound analysed (ebb and flow), we can create a realistic sound that can play all the time without repetition.

4.2.2 Grain Durations

The durations of the grains obtained with the analysis are often very contrasted. Short grains can be shorter than 1 ms, other can be longer than 100 ms.

The rules of these different grains are very different for the perception. The short grains are involved in the transient sound perception. The longer are involved in the pitch and partial perception.

Modifying the durations of the grains allows to get new sounds that are perceptively very different from the originals but with a coherent organisation. The modification can be very progressive.

You can scale the durations but more often we have used a transfer function.

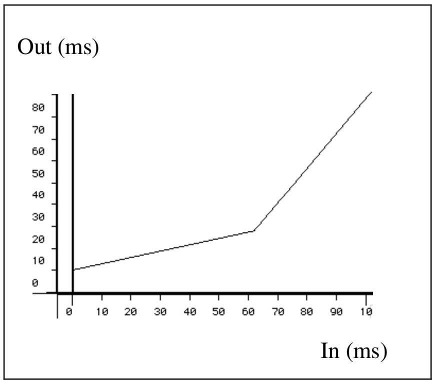


Fig. 14. Transfer function for the duration of the grains (here small durations are extended)

4.2.3 Filtering

For long grains, the frequency of the wave can be modified which allows producing a filtering effect.

We have used the Tristan Murail libraries to modify the frequencies. Two functions were specially used:

- the “dist-frq” function that can distort one chord (parameters are scale-lower, scale-higher and transpose)

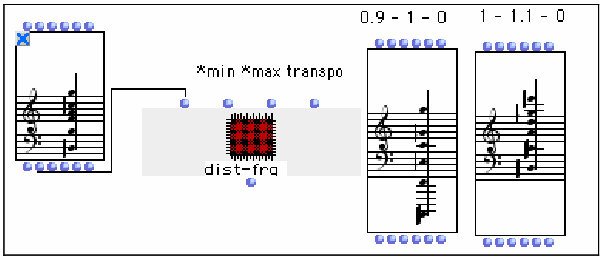


Fig. 15. “dist-frq” applied on a chord with two sets of values

- the “vocoder” function that sets all the notes of a source chord to the nearer note fund in the filtering chord.

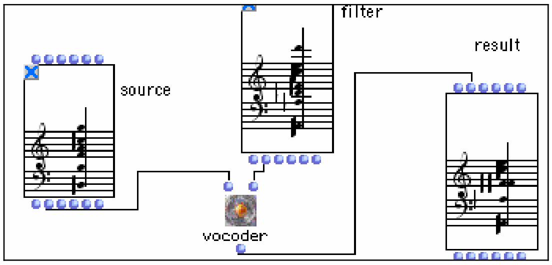


Fig. 16. “vocoder” apply a filter to a chord

We have filtered analysed sound with simple chords or with other analysis (typically Iana analysis or partial analysis). In the second case, we obtain crossed synthesis with very particular characteristics.

Tristan Murail has used these sounds for his piece "Pour adoucir le cours du temps" created in Marseille with the Prague Philharmonia, on the 21 May 2005 during the festival "Les Musiques".

4.3 SMP Analysis Synthesis in Max

The temporal domain signal model, used in MP, shows its limitations in this purpose especially with grains containing a significant stochastic part. From here comes the idea to work in the frequency domain for the decomposition process.

The principle of adaptative granular decomposition of the MP is conserved. The main idea is, rather than working with the temporal signal, to use his spectrogram for signal/atoms distance calculation.

The analysis produces very realistic sounds when the reference atoms can be well identified (this is done manually until now). We have implemented the possibility to do the synthesis with the analysis data inside Max/MSP, with the "bufgranul~" object. No musical application has been done yet. We are at the beginning of the researches on this topic but, no doubt, there is a large area to explore in this direction.

5 Conclusion

Granular synthesis is a powerful synthesis technique that can produce very different kinds of sounds. Controlling only few parameters can produce very interesting and musical variations of the sound. Inside the GMU project, we propose three connectable linkable ways of controlling parameters : graphical stochastic distributions, Composition Aided Functions for hybridizing of statistical distribution and musical scores and statistical correlated stochastic distributions extracted from natural sounds by Spectral Matching Pursuit analysis. We propose now a alpha version of the GMU software (bufgranul~ object and Max/MSP associated patches). The analysis tools have to be rewritten as a stand-alone application soon. The Computer Aided Functions will be available soon as a new library for Open-Music.

References

[1] Jean-Baptiste Barrière., Yves Potard, Xavier Rodet, "CHANT: de la synthèse de la voix chantée à la synthèse en général", Ircam publication, n°35, Ircam, Paris, 1985.

[2] Charles Bascou, Laurent Pottier, "New Sound Decomposition Method applied to Granular Synthesis", to be published in *ICMC Proceedings*, sept. 2005, Barcelone.

[3] Rémi Gribonval., Emmanuel Bacry, Stéphane Mallat, Philippe Depalle, Xavier Rodet, "Sound signal decomposition using a high resolution matching pursuit" *Proceedings of ICMC'96,* Clear Water Bay, HongKong, 1996.

[4] Todd Hodes, John Hauser, John Wawrzynek, Adrian Freed, and David Wessel, "A Fixed-Point Recursive Digital Oscillator for Additive Synthesis of Audio", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* March 15-19, 1999, pp. 993-996.

[5] Dexter Morrill, “Trumpet algorithms for computer composition”, *Computer Music Journal*, vol. 1, n°1, 1977, p. 46-52.

[6] Tristan Murail, “Spectres et Lutins”, *La Revue Musicale, special L’itinéraire*, n° 421-422-423-424, dir. D. Cohen-Levinas, Paris, 1991, p. 309—322.

[7] Laurent Pottier, “Analyse Iana - Synthèse sonore avec Csound - Contrôle avec PatchWork”, *Proceedings of the Journées d’Informatique Musicale* (JIM 94), Bordeaux, 1994, p. 74-82.

[8] Laurent Pottier, “Dynamical spatialization of sound. HOLOPHON : a graphic and algorithmic editor for Sigma1”, *Dafx98 Proceedings*, Barcelone, 1998.

[9] Laurent Pottier, “Contrôle interactif d’une voix chantée de synthèse”, *Les nouveaux gestes de la musique,* dir. H. Genevois, R de Vivo, ed. Parenthèses, Marseille, 1999, p. 175-180.

[10] Laurent Pottier, “Le contrôle gestuel de la synthèse sonore par capteurs video”, *Musiques, Arts, Technologies - Pour une approche critique*, dir. R. Barbanti, E. Lynch, C. Pardo, M. Solomos, ed. L’Harmattan, Paris, 2004.

[11] Laurent Pottier, “GMU – An Integrated Microsound Synthesis System”, *Proceedings of the Computer Music Modeling and Retrieval Conference*, Montpellier (France), May 2003, p. 98-105 ; also published in L*ecture Notes in Computer Science*, ed. Springer Verlag, feb. 2004.

[12] Curtis Roads, *Microsound*, Cambridge, Massachusetts, MIT Press, 2002.

[13] Manuel Rocha Iturbide, Gerhardt Eckel, “GIST, a granular synthesis toolkit based on an extension of the FOF generator”, *Ircam documentation*, Paris, 1995.

[14] Xavier Rodet, “Time-Domain Formant-Wave-Function Synthesis”, *Spoken Language Generation and Understanding*, ed. J.G. Simon, D. Reibel. 1980, reprinted in *Computer Music Journal* 8 (3), 1984 , pp. 9-14.

[15] Toru Iwatake, “Interview with Barry Truax”, *Computer Music Journal*, 18(3), 1994, 17-24.

Building Low-Cost Music Controllers

Alexander Refsum Jensenius[1,2], Rodolphe Koehly[2], and Marcelo M. Wanderley[2]

[1] University of Oslo, Department of Musicology, P.O. 1017 Blindern, N-0315 Oslo, Norway
a.r.jensenius@imv.uio.no

[2] McGill University, Schulich School of Music, 555 Sherbrooke Street West, H3A 1E3 Montreal, Quebec, Canada
{rkoehly, mwanderley}@music.mcgill.ca

Abstract. This paper presents our work on building low-cost music controllers intended for educational and creative use. The main idea was to build an electronic music controller, including sensors and a sensor interface, on a "10 euro" budget. We have experimented with turning commercially available USB game controllers into generic sensor interfaces, and making sensors from cheap conductive materials such as latex, ink, porous materials, and video tape. Our prototype controller, the *CheapStick*, is comparable to interfaces built with commercially available sensors and interfaces, but at a fraction of the price.

1 Introduction

A number of different sensor interfaces and sensor technologies have emerged for musical and artistic purposes in recent years, and have made it possible for "everyone" to build their own custom-made music controllers. However, most of these systems are far too expensive to allow for example all students in an undergraduate music technology class to build their own controllers, or build many self-contained instruments for a performance. This has lead to our interest in exploring various ways of creating sensor interfaces and sensors that would allow for making a complete music controller on a "10 euro" budget. Another guiding idea has been that "everyone", also people with a limited technical background, could manage to build their own controller.

The paper starts by discussing our experience with turning game controllers into generic sensor interfaces, then presents how we can make contact sensors from conductive materials, and finally shows a prototype of a music controller and how it can be used for musical applications.

2 Game Controllers as Sensor Interfaces

Commercially available sensor interfaces intended for musical and artistic purposes generally use either MIDI or Open Sound Control (OSC) to communicate with a computer (Table 1). Popular MIDI devices (e.g. iCubeX, MIDItron, Eobody) are generally cheaper, and have the advantage of being able to connect

Table 1. Comparison of a number of popular sensor interfaces for musical and artistic purposes (prices and specifications are taken from the manufacturer's websites, and have been generalized for the sake of comparison)

Product	Manufacturer	Price	Inputs	Speed	Resolution	Protocol
Pocket El.	Doepfer	€80	16	-	7 bit	MIDI
MIDItron	Eroktronix	€125	8	-	7/10 bit	MIDI
Teleo	Making Things	€130	4	-	10 bit	USB
miniDig	Infusionsystems	€330	8	100 Hz	7/14 bit	MIDI
Teabox	Electrotap	€350	8	4000 Hz	12 bit	SPDIF
GluiOn	Glui	€445	16	1000 Hz	16 bit	OSC
Eobody	IRCAM	€480	16	-	10 bit	MIDI
Wi-miniDig	Infusionsystems	€500	8	100 Hz	7/14 bit	MIDI
Digitizer	Infusionsystems	€580	32	24-244 Hz	7/14 bit	MIDI
Wise Box	IRCAM	€950	16	200 Hz	16 bit	OSC
Toaster	La Kitchen	€1200	16	200 Hz	16 bit	OSC
Kroonde	La Kitchen	€1200	16	200 Hz	10 bit	OSC

directly to any MIDI-compatible equipment. In practice, however, most people tend to connect the interfaces to a computer. Thus, a separate MIDI interface is also required, boosting both the total prize and potential problems of such a setup.

The shortcomings of MIDI in terms of resolution[1] and speed, make OSC based sensor interfaces (e.g. Ethersense, Toaster, GluiOn) more attractive. Such devices typically allow for higher resolution and sampling rates, and long-distance communication over standard high-speed ethernet connections [1]. There are even devices that use digital audio for communication (e.g. Teabox [2]), but this requires the computer to be equipped with a digital audio input.

The problem is that none of these devices come close to our "10 euro" budget. Most of them are also too bulky for our needs, since we are interested in integrating as much as possible on the music controller itself. A solution, of course, might be to use microcontrollers such as the Atmel, Pic or Basic Stamp, as suggested by [3, 4], but the background in programming and electronics needed to succeed with this is not something we expect from the regular music student taking a music technology class.

Using consumer electronic devices such as game controllers then seems like a better option. They are easily available for a low prize at any electronics store, and work out of the box using the generic Human Interface Device (HID) driver available with most operating systems [5], and supported in programs such as Max/MSP, PD and Matlab. This means that no extra interfaces are needed and no software has to be installed to make them work. Since they also draw all their power from the USB-port, no external power supply is needed either.

[1] Usually 7 bit (0-127), although some products allow for sending dual messages giving a 14 bit range.

Of the many different types of game controllers available, we find gamepads the most interesting. They are generally cheaper and smaller than other types of controllers (such as joysticks, flightsticks or wheels) and often have the largest number of inputs. Taking a typical gamepad apart reveals a motherboard with 4 analog and 12 digital inputs. These inputs comply to the standard 0-5 volt sensor input range, and thus most sensors can be connected to the input points on the main board directly. In most cases the motherboards are clearly marked, so it is easy to see where the different connectors are, but in some cases it might be necessary to check with a multimeter which connectors carry signal, +5 volt and ground. For simplicity and for making a generic sensor interface, we usually de-solder the small joysticks, and solder on cables with 3-pin connectors. This makes it is easy to test different types of sensors with the interface.

The result is a generic sensor interface with 4 analog and 12 digital inputs, 8-bit resolution and 100Hz sampling rate. Using a “rumblepad” will even give a couple of analog outputs and small motors, and a wireless gamepad can be turned into a wireless controller. For large projects with a need for lots of inputs, it is possible to connect several controllers through USB-hubs.

3 Sensors

For our musical applications, we are mostly interested in contact sensors that allow for the same type of sensitivity and tactile feel as acoustic instruments. A problem with many commercially available products, such as bend, pressure and position sensors, is that they ship only in standard sizes and defined materials [6]. This constrains the possibilities of the controller to the shape and size of the available sensors, and draws the focus away from the musical applications.

Another problem with commercial sensors is the lack of tactile feel and response, either because they are too thin and small, or they are too big and have too low resolution to be interesting for musical purposes. These problems can be overcome by adding extra material around the sensors, but this also reduces the sensor’s response since the padding effectively filters the gestural energy.

This is not to forget that the prize of only one commercial sensor is often many times our “10 euro” budget, and in most cases it is necessary to have a number of different types of sensors on a controller. But rather than looking at adding lots of sensors to increase the response of a controller, a solution might be to use the sensing material itself as a transducer. For example, for a “shoe controller” the sole of the shoe could be the sensor itself, rather than adding various sensors to a regular sole such as in [7]. Ideally, conductive materials could be bought for example in rolls and then be adapted to the desired thickness and sensitivity. This would also make it possible to adjust the sensitivity of the sensor to the envisioned usage, for example high resolution for a hand controller and lower resolution for a foot controller. Thus, any form and dimension could be realized with the same technology.

Such conductive materials already exist, for example artificial piezoelectric polymers (PolyVinyliDene Fluorid, PVDF) tested in several musical interfaces such as the Magic Carpet [8], or the electret, a porous conductive polymer film used for pick-ups in some acoustic instruments. They are, however, quite expensive and since they are mass-produced they are only available in certain standard sizes from the manufacturers. Other surface conductive materials, for example some types of video tape, can be used directly for making sensors [9, 10]. Another low cost alternative is to make new materials mixed of conductive parts and some non-conductive or low-conductive and highly visco-elastic material, usually some polymer, such as the plubber, a pressure sensitive silicon/carbon composite used in the z-tile [11].

We have succeeded in making various types of position, pressure and bend sensors from components generally available in hobby or artwork stores, including conductive ink, adhesive, rubber, tape, elastics and porous materials. Since such materials can be bought in many different sizes, it makes it easy to customize the sensing surface to the interface we are interested in building. As can be expected, the conductive qualities of these materials varies quite a lot, so further research is needed to test different types of materials and improve the consistency and reliability of the response. Also, since we are working with materials where the range and conductivity is unknown, more work will also need to be put into improving the electronic circuits for optimizing the signals to match the 0-5 volt range used in the sensor interfaces.

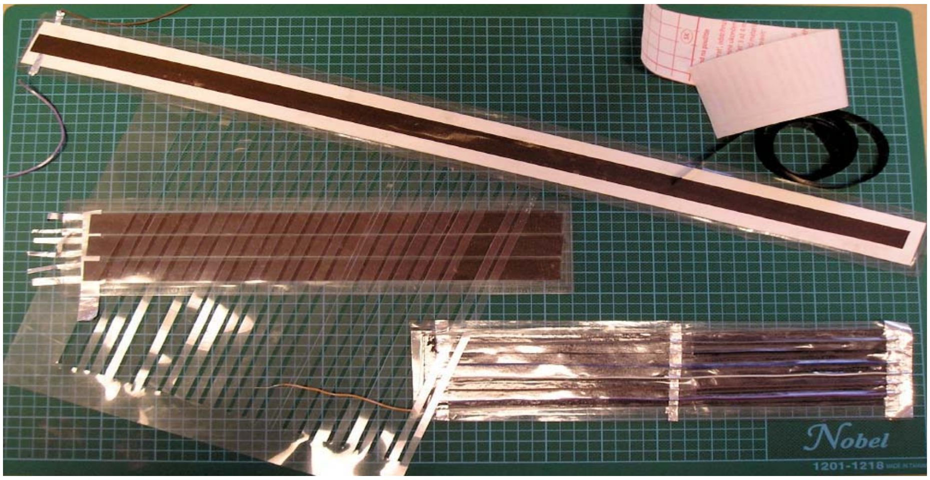


Fig. 1. Various sensors made from conductive materials

4 A Prototype Musical Controller

To test how our interfaces and sensors work in a musical context, we built the *CheapStick*, a simple prototype controller with three pressure sensors made from a porous material and a position sensor made from video tape (Figure 2, 3). With the sensor interface mounted on the board, this makes a self-contained music

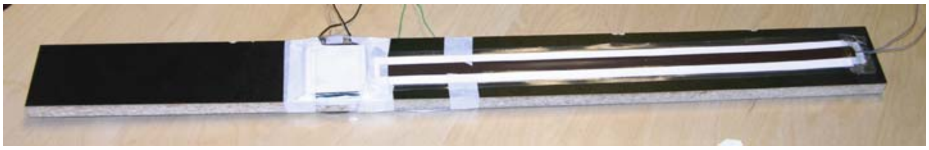


Fig. 2. The *CheapStick* was built with pressure sensors made from a porous material and a position sensor made by video tape

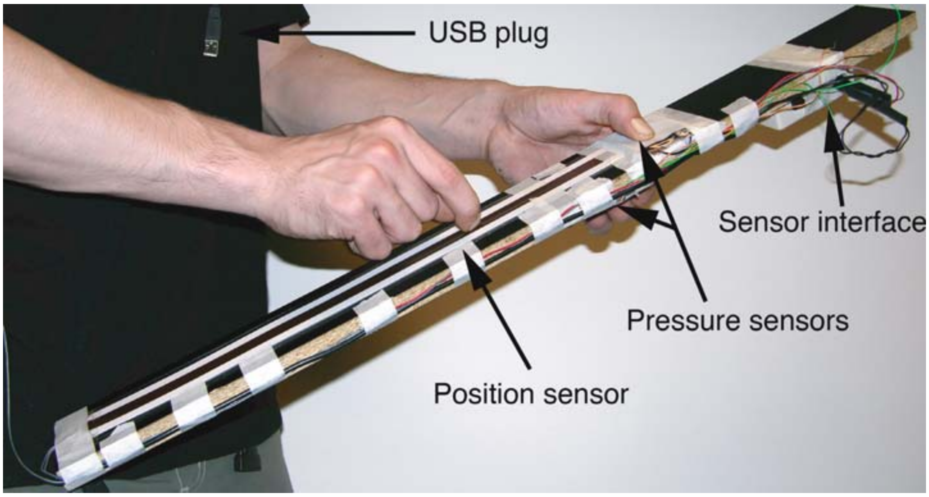
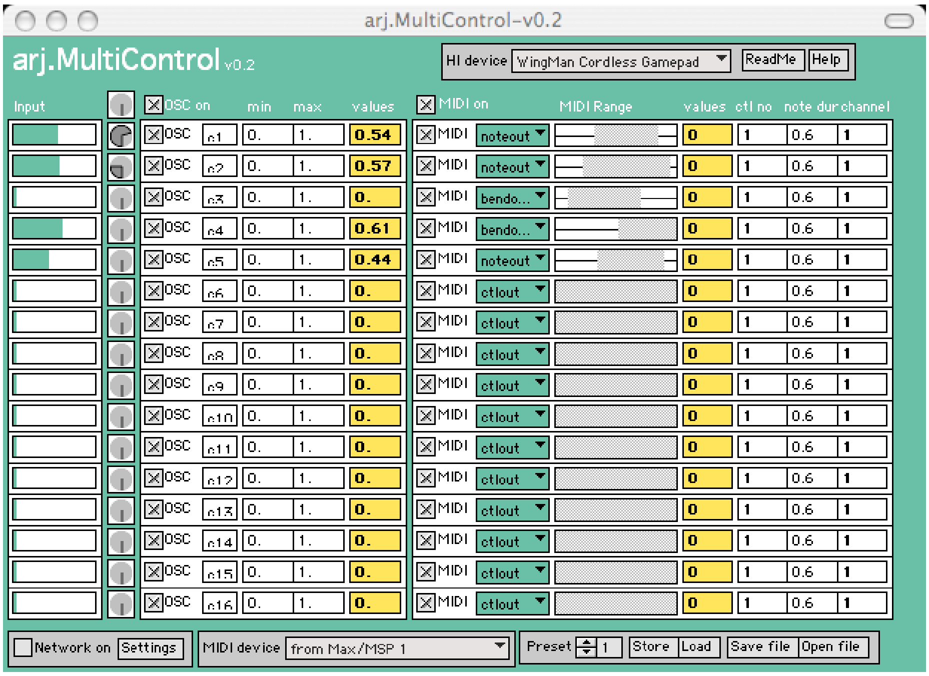


Fig. 3. The *CheapStick* can be played using the right hand to control position, and the left to control the pressure sensors

controller very close to our "10 euro" budget, and with a USB-plug that can easily be connected to a computer and used with various music applications.

HID-compliant devices can be accessed directly with the HI object in Max/MSP or the HID object in PD. To help in making an easy-to-use setup for musical applications, we have made the *MultiControl*[2] program (Figure 4). This program is somewhat similar to the MIDI-based Junxion [12], but adds a number of useful features. The program can access any HID-compliant controller connected to the computer and will automatically detect the active channels from the device. The incoming values are automatically scaled to a usable range of floating point numbers between 0 and 1, and these values can also be smoothed which might be useful if the sensors send a noisy signal. The program can output either Open Sound Control (OSC) messages internally or on the network, or output MIDI so that it can be used with any MIDI-compatible equipment. This program makes it easy to experiment with different mappings, and test out the musical possibilities of the controller.

[2] Available from http://musicalgestures.uio.no


arj.MultiControl-v0.2

arj.MultiControl v0.2

HI device: WingMan Cordless Gamepad

☒ OSC on		min	max	values	☒ MIDI on		values	ctl no	note dur	channel
☒ OSC	c1	0.	1.	0.54	☒ MIDI	noteout	0	1	0.6	1
☒ OSC	c2	0.	1.	0.57	☒ MIDI	noteout	0	1	0.6	1
☒ OSC	c3	0.	1.	0.	☒ MIDI	bendo...	0	1	0.6	1
☒ OSC	c4	0.	1.	0.61	☒ MIDI	bendo...	0	1	0.6	1
☒ OSC	c5	0.	1.	0.44	☒ MIDI	noteout	0	1	0.6	1
☒ OSC	c6	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c7	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c8	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c9	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c10	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c11	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c12	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c13	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c14	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c15	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1
☒ OSC	c16	0.	1.	0.	☒ MIDI	ctlout	0	1	0.6	1

☐ Network on | MIDI device: from Max/MSP 1 | Preset 1

Fig. 4. Screenshot from the MultiControl software which allows for smoothing and scaling of controller data, and output to OSC or MIDI

5 Conclusion and Future Work

We have presented our current work on turning consumer game controllers into generic sensor interfaces, and making sensors from various conductive materials. This "10 euro" controller performs quite similar to systems that cost many times this price. Although initially motivated by cost and easiness rather than high quality, speed and resolution, we are impressed by the tactile feel and response of some of our low-cost sensors.

References

1. Emmanuel Fléty, Nicolas Leroy, Jean-Christophe Ravarini, and Frédéric Bevilacqua. Versatile sensor acquisition system utilizing network technology. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME04), Hamamatsu, Japan*, pages 157–161, 2004.
2. Jesse Allison and Timothy Place. Sensorbox: Practical audio interface fo gestural performance. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03), Montreal, Canada*, 2003.
3. Dan Overholt. The create usb interface - where art meets electronics. http://www.create.ucsb.edu/ dano/CUI/, 13 November 2005.

4. Dan O. Sullivan and Tom Igoe. *Physical Computing - Sensing and Controlling the Physical World with Computers*. Thomson Course Technology, Boston, MA, 2004.
5. http://www.usb.org/developers/hidpage, 13 November 2005.
6. Marcelo M. Wanderley and Rodolphe Koehly. Methods for the in-house development of sensors for musical applications. Manuscript in preparation.
7. Joseph Paradiso and Eric Hu. Expressive footwear for computer-augmented dance performance. In *Proceedings of the First International Symposium on Wearable Computers, Cambridge, MA*, pages 165–166. IEEE Computer Society Press, 1997.
8. Joseph Paradiso, Craig Abler, Kai-yuh Hsiao, and Matthew Reynolds. The magic carpet: Physical sensing for immersive environments. In *Proc. of the CHI '97 Conference on Human Factors in Computing Systems, Extended Abstracts*, pages 277–278, NY, 1997. ACM Press.
9. T. Escobedo. The synthstick. http://www.geocities.com/tpe123/folkurban/ synthstick/synthstick.html, 7. May 2005.
10. Andrej Stordeur. Ribbon controller. http://www.angelfire.com/music2/ theanalogcottage/ribcont.htm, 8. March 2005.
11. Lisa McElligott, Michelle Dillon, Krispin Leydon, Bruce Richardson, Mikael Fernstrom, and Joe Paradiso. *UbiComp 2002, LNCS 2498*, chapter 'ForSe FIElds' - Force Sensors for Interactive Environments, pages 168–175. Springer-Verlag Berlin Heidelberg, 2002.
12. Steim. Junxion (computer program). http://www.steim.org/steim/junxion.html, 8. March 2005.

Evaluation of Sensors as Input Devices for Computer Music Interfaces

Mark T. Marshall and Marcelo M. Wanderley

Input Devices and Musical Interaction Laboratory,
McGill University - Music Technology,
555, Sherbrooke St. West,
Montreal, QC, Canada
mark.marshall@mail.mcgill.ca, mwanderley@music.mcgill.ca
http://music.mcgill.ca/musictech/idmil/

Abstract. This paper presents ongoing research into the design and creation of interfaces for computer music. This work concentrates on the use of sensor as the primary means of interaction for computer music, and examines the relationships between types of sensors and musical functions. Experiments are described which aim to discover the particular suitability of certain sensors for specific musical tasks. The effects of additional visual feedback on the perceived suitability of these sensors is also examined. Results are given, along with a discussion of their possible implications for computer music interface design and pointers for further work on this topic.

1 Introduction

The use of sensor technology is a fundamental part in the creation of interfaces for computer music. However little investigation has taken place into the suitability of particular sensors for specific tasks in these interfaces. While a number of taxonomies and evaluations of sensors have taken place [1] [4], these have not been concentrated on the use of such sensors in musical applications.

The experiments described in this paper have been designed to investigate a number of important aspects in the use of sensor technologies in these interfaces. This includes investigation of the usability of particular sensors for specific musical tasks and investigation of the effects of additional visual feedback on this usability.

The work performed has involved three major phases. These are:

– a survey of existing interfaces for computer music and the use of sensors in them
– classification of the sensors based on the parameters sensed
– experiments to determine the suitability of sensor classes for musical tasks

This document will discuss each of the phases of the research, along with the results achieved and the possibilites for future work evident from these results.

2 Survey of Sensor Use in Interfaces for Computer Music

In order to allow the results of our experiments to be as useful as possible, it was determined that the sensors examined should be representative of those most commonly found in computer music interfaces. To facilitate this a survey was made of a large number of computer music interfaces to determine which sensors were the most common.

2.1 Scope of the Survey

The first step in the survey involved determining where to find the information related to the instruments in order to be able to detemine which sensors were used in them. It was decided to use the instruments which had been presented at a major conference on the design of digital musical interfaces as the basis for the survey. Therefor we examined the instruments presented at the New Interfaces for Musical Expression (NIME) conferences from 2001 to 2004, which resulted in a total of 123 instruments and interfaces being surveyed.

It should be noted that the survey only examined the sensors used, so that complex devices such as joysticks and cameras were ignored. This reduced the number of interfaces to 105, due to 18 interfaces which were controlled solely by means of a complex device. A further 54 interfaces contained a combination of complex devices and sensors.

The ten most used sensors based on this survey are shown in Table 1 (from most often to least often used). It should be noted that this count is based on the number of distinct instruments using the sensor (i.e. while 19 instruments use accelerometers, each of these instruments may have used more than one accelerometer as well as other sensors). There are also a further 12 sensors which were present in 4 or less instruments and are not shown in the table.

Table 1. Most commonly used sensors

Sensor	Number of Instruments
Accelerometer	19
Force Sensing Resistor	18
Infrared Sensor	9
Light Sensor	8
Touch Pad	8
Bend Sensor	6
Capacitive Sensor	6
Rotary Potentiometer	6
Gyroscope	5
Linear Potentiometer	5

3 Categorisation of Sensors and Tasks

Previous work has attempted to show a mapping between sensors and classes of musical task [5]. This work classified sensors by the form of input that they sensed (linear position, rotary position, force etc.) and classified musical tasks by the range and form of input they required (static, absolute dynamic and relative dynamic). The experiments described here make use of these categorisations and attempt to evaluate empirically whether any mapping from sensor type to musical task holds.

Table 2 shows how the sensors used in this experiment were classified. The sensors chosen are all among the ten most commonly used sensors as found by our study and have been selected to allow for at least one sensor from each class.

Table 2. List of sensor devices used in the experiments and their associated categories

Sensor	Sensor Category
Linear potentiometer (fader)	Linear position
Rotary potentiometer	Rotary position
Linear position sensor (ribbon controller)	Linear position
Accelerometer	force
Force sensing resistor	force
Bend sensor	rotary position

The task list consists of two simple tasks and one complex task, the complex task being created by combining the two simple tasks. The two simple tasks have been chosen to represent common musical tasks, while also conforming to the classification of musical function as presented in [5]. It has been proposed in [7] that the concept of a musical task is an inherent part of the evaluation of controllers for computer music. The authors also presented a partial list of musical tasks which might be used in the evaluation of controllers for computer music. The tasks chosen for this work are based upon this list and have been categorised based on the scheme presented in [5]. The tasks are also consistent with those used in [6] to allow for a comparison between the results of these experiments and those of that work.

Table 3 shows the chosen tasks and their classification.

Table 3. List of tasks and their associated categories

Task	Task Category
Note selection	Absolute dynamic
Note modulation	Relative dynamic
Note selection & modulation	Complex combined

4 Test Setup

Two experiments have been designed, each of which involves performing the selected tasks with each of the selected sensors. The experiments differ only in the feedback provided to the user during the tasks. The experiments themselves will be described in detail in later sections.

The setup is the same for each experiment. It consists of the user manipulating a synthesis system through the use of a sensor and a button. Pressing the button causes a sound to be emitted from the system, the frequency of which is controlled by the sensor. The sensor is controlled with the user's primary hand, the button with the secondary hand.

Synthesis is performed in Max/MSP and is a simple waveshaping synthesis system based on Chebychev equations. The frequency of the synthesis is variable in semitones.

There was a total of 11 participants in the test group. The participants were all graduate students in Music Technology and their areas of specialisation ranged from acoustics and physical modelling to interaction design to music information retrieval.

Eight of the participants had extensive musical instrument training, while the remainder either did not play, or had only played for a period of less than two years and had since stopped. Five participants had experience of playing electronic instruments whether software or hardware in form.

As already stated, each experiment consisted of three tasks. Each task was performed with each sensor. When a task had been completed with all sensors a short break was taken before beginning the next task. A task was considered to be completed with a given sensor when the user was happy that they had performed it as well as they could, or when the user decided they could not perform the given task with the particular sensor.

Information from the experiments was gathered by a number of means. On completing a given task with a sensor, the user was asked to rate the ease of use of this sensor for the task, by setting the position of a slider object in Max/MSP. This slider gave a percentage rating of the ease of use of the sensor, which was recorded. This allows us to gather data indicating the subjective usability of the sensors for the tasks.

The length of time taken before completion of the task was noted, along with the success or failure of the task. This gives an indication of the learnability of the sensor, as well as its suitability for the task.

A video recording containing the interaction with the system and the audio from the system and user themselves was also made. This allowed for later analysis of factors such as ease of learning, accuracy and quality of sound produced with each sensor.

Finally, the users were debriefed verbally after each task was complete and asked to comment on any particular strengths and weaknesses of the sensors for that task. This gave a subjective opinion of the sensors as well as offering ways of possibly improving the interaction of a given sensor.

5 Experiment 1 – No Additional Feedback

5.1 Description

The first experiment is the baseline experiment for determining the suitability of the sensors for the tasks. It consists of the user manipulating the frequency of synthesis with the sensors and causing a sound to be emitted with a button. No additional feedback (haptic, tactile or visual) is given from the system except for that intrinsically provided by the sensor itself.

5.2 Tasks

The first task is the note selection task. The user is asked to attempt to play a short melody with each sensor. Choice of the melody is left to the user. The sensor can be used to manipulate the pitch of the sound produced in intervals of a semitone, with a range of one octave. The note currently selected by the state of the sensor is emitted when the button is pressed. The users are asked to restrict themselves to emitting short single notes. The task is considered completed when the user feels they have played the melody to the best extent allowed by the sensor.

The second task is a note modulation task. In this task the computer plays a short melody, one note of which is emitted at every button press. Notes can be sustained by holding the button. The user is asked to play the melody through, sustaining every fourth note and adding a trill effect between this note and the note above it using the sensor. Thus the sensor is used to modulate between the current semitone and the next.

The third task is a combination of the first two. The users are again asked to play a short melody, but as well as the short single notes used in the first task, they are now allowed to sustain and modulate notes using the sensor. This provides a more complex task than the previous two and allows us to examine the effects of increased task complexity on the sensors suitability.

5.3 Results

For the first task, note selection, the users showed a very strong preference for the linear position sensor. The linear and rotary potentiometers were next in preference and received similar ratings.

For the note modulation task user preference was split between the linear position sensor and the force sensing resistor. Viewing of the recorded video from the experiment indicated that preference was highly dependant on the technique used to manipulate the sensors. All users giving preference to the force sensing resistor attempted the modulation using the linear position sensor by sliding their finger along the sensor. Those prefering the linear position sensor used two fingers in a rocking motion, similar to playing a trill on a keyboard. It should be noted that those who performed the sliding movement were creating a more vibrato-like effect and that the preference of these participants is consistant with the results of [6] who found that users prefer the force sensor for creating vibrato effects.

Finally, for the complex task, user preferences seemed to depend on their preferences for the first two tasks. Sensors which were preferred for part of the task (i.e. for one of the simple tasks) were rated well for the whole task. The linear position sensor and force sensing resistor were the prefered sensors.

Figure 1 shows the average normalised ratings for each sensor for each of the tasks. These ratings were achieved by normalising each users rating relative to the highest rating they gave and then finding the mean of these ratings across users.

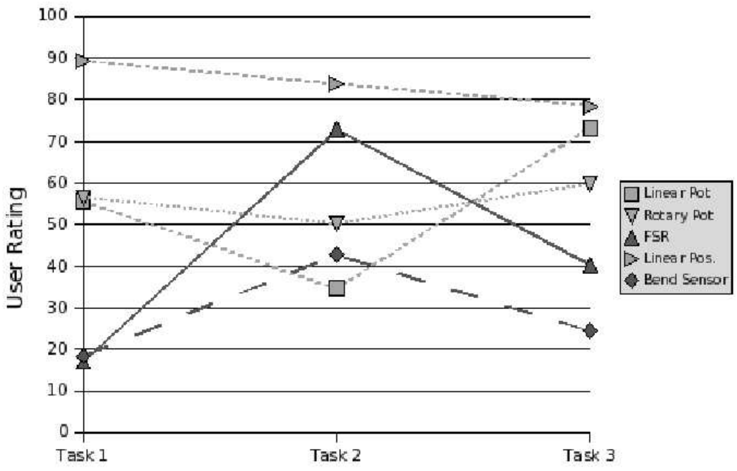


Fig. 1. Normalised average user ratings for each sensor for each task

6 Experiment 2 – Additional Visual Feedback

6.1 Description

Evidence exists that tactile and kinaesthetic feedback prove important to expert musicians playing traditional instruments ([3]). However, it has been stated in [5] that visual feedback is most useful to beginning musicians. Therefor, the second experiment performed consisted of the same setup and tasks as the original experiment, but with the addition of a visual feedback system. This visual feedback system involved the displaying of a line of white boxes on the screen. Each of these boxes represented a semitone over the octave range of the sensor. The semitone which was selected by the current value of the sensor was highlighted by a yellow circle within the box representing that semitone. The boxes were 2.5 x 2.5 cm in size, and were displayed against a grey background.

6.2 Tasks

As stated previously, the tasks chosen for this experiment are the same as those in the first experiment. Therefor, the participants perform a note selection task (playing a melody), a note modulation task (adding a trill to an automatic melody) and a composite task (playing a melody with added trills). The difference between these experiments is solely due to the additional visual feedback.

6.3 Results

For the note selection task and the combined task, with the exception of the FSR and the linear position sensor, all sensors showed a large increase in the normalised rating given to them by each user. Each of these sensors achieved a rating at least 25% higher than without feedback. The FSR improved by only 0.4% and the linear position sensor showed a decrease in rating of 4%. Comments from users about the linear position sensor indicate that the difference in location of the visual feedback system and the sensor itself causes confusion about which one to pay attention to. It is possible that were the visual feedback system integrated into the sensors, this confusion would not arise. Figure 2 shows the user ratings for each sensor for this task, both with the additional feedback and without.

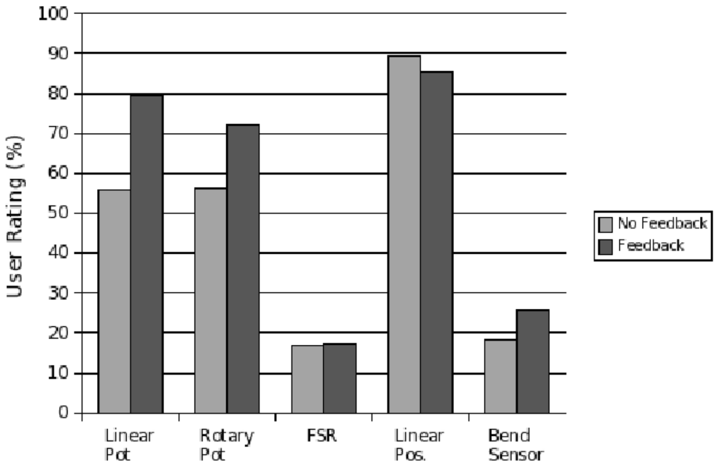


Fig. 2. User ratings for each sensor both with and without visual additional feedback

Also for these two tasks, a major improvement was also found in the accuracy of users once the visual feedback system was added to the experiment setup. An improvement in accuracy of at least 8% was found in all sensors not belonging to the linear position class of sensors. This is shown in Figure 3. When these improvements are taken relative to the accuracy achieved without the additional feedback, this shows a minimum relative improvement of 15% for these sensors, as shown in Figure 4.

The results of the modulation task are not examined here as interviews with the participants indicated that the majority of them were not using the visual feedback for this task. They found that it offered no advantage to use it and so did not.

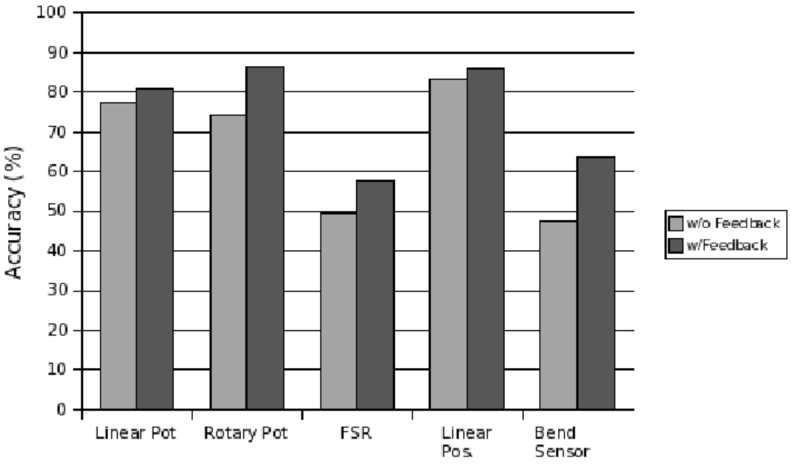


Fig. 3. User accuracy for the note selection task, both with and without additional feedback

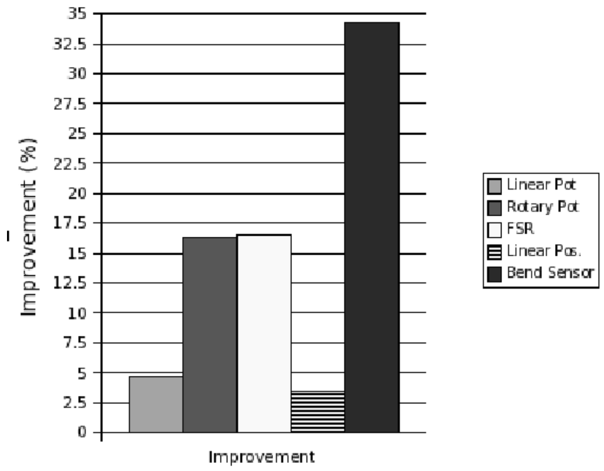


Fig. 4. Improvement in user accuracy for each sensor, relative to their initial accuracy

7 Overall Results

As can be seen from the results of experiment 1, users show a strong preference for certain sensors for specific tasks in musical insterfaces. These preferences are consistant across many users, with the only obvious variation in preference (some users preferring the FSR to the linear position sensor for the note modulation task) being explainable by the technique used in performing the task.

Also notable is the effect of the additional visual feedback in experiment 2. User accuracy in the note selection task was greatly improved, with the majority of users now capable of playing a melody with all sensors. This indicates that proper visual feedback in an instrument system can greatly increase the playability of the system. It is interesting to note that the positive effect of this visual feedback (which was in the form of a linear representation of the notes) was only present for sensors which are not linear position sensors themselves. This may be due to the linear visual feedback which is inherent in the linear position sensors.

8 Conclusions

This paper presented the results of a number of experiments to determine the suitability of sensors for specific tasks in digital musical instruments and the effect of the addition of visual feedback on this suitability. The experiments have shown that users do express a preference for certain types of sensor for certain musical tasks and that these preferences are consistant across users. The results for users producing a vibrato-style modulation in the modulation tasks also proved consistant with those of previous work in this area [6].

Also shown was that additional visual feedback had an effect, not only on the perceived suitability of the sensors for the tasks, but also on the accuracy of the users when using the sensors. These results show that it should be possible to derive guidelines for the use of sensors in digital musical instrument interfaces and for the use of visual feedback to improve the interaction in these instruments.

It is hoped that these experiments will aid in the future design of computer musical instruments, by providing an indication of the mappings suitable for a particular sensor or for a particular parameter in an interface. By careful choice of the sensors and mappings used in an instrument interface, instruments more suited to expert performance can be created [2].

Acknowledgements

We would like to thank all of the participants who took part in this study, for taking the time to participate in the experiments and for their valuable comments.

This work is supported by the Natural Sciences and Engineering Research Council of Canada, through a Discovery Grant to the second author and by the Centre for Interdisciplinary Research in Music Media and Technology, through a student support grant to the first author.

References

1. Fraden, J.: Handbook of modern sensors, Springer-Verlag, Heidelberg, Germany (2000)
2. Hunt, A., Wanderley, M.M., Kirk, R.: Towards a model for instrumental mapping for expert musical performance In: Proceedings of the International Computer Music Conference, Berlin, Germany (2000)
3. Keele, S.W.: Attention and Human Performance Goodyear Publishing Company (1973)

4. Pallas-Areny, R., Webster, J.G.: Sensors and Signal Conditioning, 2nd Edition John Wiley & Sons Ltd. (2000)
5. Vertegaal, R., Ungvary, T., Kieslinger, M.: Towards a musicians cockpit: Transducers, feedback and musical function In: Proceedings of the International Computer Music Conference 308-311, Hong Kong (1996)
6. Wanderley, M.M., Viollet, J., Isart, F., Rodet, X.: On the choice of transducer technologies for specific musical functions. In: Proceedings of the International Computer Music Conference, Berlin, Germany (2000)
7. Wanderley, M.M., Orio, N.: Evaluation of input devices for musical expression: borrowing tools from HCI. Computer Music Journal **26(3)** 62-67 (2002)

Aspects of the Multiple Musical Gestures

Kristoffer Jensen

Aalborg University Esbjerg, Niels Bohrsvej 6, 6700 Esbjerg, Denmark
+45 79 12 76 66
krist@cs.aaue.dk

Abstract. A simple to use pointer interface in 2D for producing music is presented as a means for real-time playing and sound generation. The music is produced by simple gestures that are repeated easily. The gestures include left-to-right and right-to-left motion shapes for spectral envelope and temporal envelope of the sounds, with optional backwards motion for the addition of noise; downward motion for note onset and several other manipulation gestures. The initial position controls which parameter is being affected, the notes intensity is controlled by the downward gesture speed, and a sequence is finalized instantly with one upward gesture. Several synthesis methods are presented and the control mechanisms are mapped into the multiple musical gesture interface. This enables a number of performers to interact on the same interface, either by each playing the same musical instruments simultaneously, or by performing a number of potentially different instruments on the same interface.

1 Introduction

The common understanding of a musical instrument is that it controls the note pitch, length and dynamics to some degree, and furthermore has some possibilities of changing the timbre, most noticeable in the singing voice and less in other acoustic instruments. In modern music, even less control is often obtained, because of the use of pre-sampled sequences in the creation of the music.

The actual control structure of digital instruments, which is most often MIDI-based, does not easily permit more continuous control of an instruments timbre, by its inherent note-onset structure. Whereas this is indeed an appropriate control structure in many situations, the lack of sound timbre control can be disadvantageous in other situations.

The continuous control of the timbre is often, in today's computer-based music tools, replaced by short sequences that are concatenated to produce pleasing music, often with standard rhythmic structure. While this method shows the need for the easy production of music with standard orchestration and rhythm, it also pin-points the popularity of such sequences in today's popular music. As for the continuous control of timbre versus note-onset control structure, the automatic approach will become more popular, and research into appropriate interaction modalities is necessary to permit the easy manipulation and creation of sounds and music sequences in real-time.

Music synthesis has developed a great deal since the advents of the first FM synthesis methods [1]. While the gain in instrument sound fidelity is unquestioned, the use of pre-recorded samples often prevents an easy real-time transformation of the timbre. In contrast, more parametric models, for instance the additive synthesis method with appropriate high-level parameterization, such as the timbre model [3], can permit timbre changes easily. With the novel parameterized synthesis methods, there is a need for appropriate, flexible and intuitive interfaces.

While the advent of flexible music programs widens the scope of musical sounds, it also limits the musical interaction in many situations. The performance of a musical sound is a vital step in music production by the addition of crucial timing and dynamics [8]. Even more so, the simultaneous performance of several instruments additionally contains complex synchronization issues that both heighten the music quality, but that also potentially alter the music in new directions.

This paper presents three sound models in section 2, maps the sound model parameters to the multiple musical gesture in section 3 for several cases, gives information about potential implementations in section 4, and a conclusion is given in section 5.

2 Sound Models

As the multiple musical gesture are supposed to affect several important timbre attributes, an expressive and versatile sound model is necessary. The timbre model, a high-level additive model, is such a model. It has been shown to synthesize many acoustic instruments with good fidelity [3], but it also renders a large variety of irregular and noisy sounds [10].

In contrast, the interface is expected to have several, if not many sounds playing at the same time. This, in combination with the interface algorithms, put a heavy load on the host computer. For this reason, two relatively cheap, albeit powerful algorithms are used to produce the sounds that constitute the music; one is a traditional harmonic synthesis with perceptual relevance, the other an unvoiced synthesis method that creates anything in between Geiger (clicks) or cymbal (inharmonic tones) sounds.

2.1 The Timbre Model

A particular implementation of the additive synthesis, the Timbre Model [3], is chosen as the sound model first implemented in this work. In this model, a number of sinusoids (pure tones) with quasi-harmonic frequencies and time-varying amplitudes are made less pure by adding band-limited noise on the frequencies or amplitudes of the sinusoids. The amplitude and spectral envelope are not defined in this work, as they are controlled by the performer. In order to play the sound with an arbitrary duration, the sound is scaled using segment knowledge. Given an amplitude envelope, the segments are found using the derivatives of a smoothed envelope [4], or by identifying the first 90% of the maximum as the end of attack, and the last 70% of the maximum as the start of release. Only the decay/sustain part of the sound is scaled when controlling the synthesis. The sustain part is not affected, as it has similar start and end amplitudes, while the decay amplitude is decreased by a given dB value per second, according to the gestured decay curve.

The sound of the timbre model is created by summing a number of sinusoids. The amplitudes and frequencies are made by summing a filtered noise with the static values. The noise is a scaled sum of a common and an individual component for each partial.

The control structure of the timbre model consists of the individual static amplitudes and frequencies, and the irregularity parameters. The irregularity parameters are also individual for each partial, and they consists of strength (std), bandwidth (BW), and correlation, which is how strong the noise part of the sound is, how rumbling versus hissing it is and how much the irregularity of a partial is similar to the irregularity of the fundamental, respectively. A further development of these irregularity parameters can be found in [10]. A special separation of the irregularity is attempted in this work. By splitting the irregularity into two frequency bands, one low-frequency (below approximately 10 Hz) and one high-frequency (up-to approximately the fundamental frequency of the note played), an intuitive control can be made without involving the bandwidth of the filter. This permits the independent control of either the rumbling or the hissing quality of the sound.

2.2 Brightness Creation Function

In case many notes are to be produced, a more cost-effective synthesis method is needed. The brightness creation function formula, known from many mathematic formula books, was first put into musical relevance by Moorer in 1976 [1]. Jensen [3] showed that this formula had an easy link to the spectral centroid, which in turn is closely linked to the perception of brightness. The brightness creation formula (BCF) is,

$$s(t) = a_0 \frac{1}{\pi} \cdot \frac{B\cos(\omega_0 t) - 1}{B^{-1} + B - 2\cos(\omega_0 t)}, \tag{1}$$

where a_0 is the amplitude of the sound, ω_0 is the fundamental frequency, and B as a function of the spectral centroid T_b is given as,

$$B = \frac{T_b}{T_b - 1}. \tag{2}$$

Thus, by a simple formula, a sound is created that allows one to control the amplitude (linked to the perceptual loudness), fundamental frequency (pitch) and length of the sound. In addition, the principal timbre attribute, the spectral centroid (brightness) is also controlled. By putting an envelope [4] on the amplitude, other important timbre attributes, such as the attack time, decay rate, etc. are easily created. As with the timbre model, only the decay/sustain part of the sound is time-scaled when controlling the synthesis. The sustain values are not affected, while the decay amplitude is decreased by a given dB value pr second. Finally, by adding a band-limited noise on the amplitude and fundamental frequency, several additional timbre attributes are controlled, having to do with irregularity, noise component, etc. Noise on the amplitude is called shimmer, and noise on the frequency, jitter. The synthesis method has much in common with the timbre model, although the different timbre attributes are only controllable for all harmonics together, not individually.

2.3 Atomic Noise

The atomic noise [5] is a synthesis method for unvoiced sounds that can create a large variety of sounds with subtle variations. By adding a variable number of atoms with random center frequency and width, different noise categories are created, including Geiger (clicks) and cymbal (inharmonic tones). The formula for one atom is,

$$a(t) = a_0 \cos(2\pi\omega_0 t)e^{\left(\frac{t-t_0}{\sigma}\right)^2} \text{, } if \quad r > p \text{ ,} \tag{3}$$

where a_0 is the amplitude, ω_0 is the center frequency, t_0 is the center time, and σ is the standard deviation (width) of the atom. The atom is realized at time t_0, if the random variable r is greater than the probability threshold p. The atomic noise is the sum of a number of atoms. All of the parameters are random variables with uniform or Gaussian distribution. Recent additions to this model are the loudness dependent inverse auditory filter, and a frequency distribution that renders approximately the same number of atoms in each critical band. These additions create a perceptual, instead of a physical white sound.

The distribution parameter that is controlled is the range, the maximum value in the uniform case, or the standard deviation in the Gaussian case. The amplitude range is not changed, as this would change only the perceived loudness. It is possible to change the distribution of the frequency and time to obtain, for instance, a voiced sound, by making a periodic probability density function of the time or frequency random variable [5]. What are controlled are the probability (p), the width of the atom (σ), the auditory system loudness dependent inverse filter and the frequency and time periodicity and period. By setting a low probability, very few atoms are realized, creating either a Geiger sound, if σ is small, or a cymbal sound, if σ is large. By increasing the probability, more dense sounds are created, resulting finally in a white noise, if the combined p and σ values are large enough.

3 Multiple Gestures Interface

In the multiple musical gesture interface [6], which can use any pointing device, a pointer is entering a valid square from different directions of incidence, thereby controlling either sound or note parameters. It is possible to profoundly modify the sound in the same interface as the notes are played.

The actual control is dependent on many things. First, of course, the actual sound model may or may not have the necessary parameter control. In [6], the sound model, the timbre model [3], has individual control of amplitude, frequency and irregularity on each partial. The note model was a standard note-onset model, with dynamics and vibrato control. The actual pointer device also influences the multiple musical gesture interface. For instance, the presence or not of an on/off switch could alter the interaction significantly. Such a switch is assumed to exist here. Another important aspect to be considered is the performer skills. If a novice is to use the interface, continuous pitch may be translated into the discrete notes of the major scale, the continuous temporal envelope into a simple attack/decay/release envelope, etc. The increased complexity of the interface is thus coupled to an estimation of the skill of the performer.

3.1 Timbre Model and BCF

While the multiple musical gesture interface is indeed dependent on both the sound model, the pointing device and the skill of the performer, care has been taken to insure that it has as many commonalities as possible. This includes many of the sound, note and timbre gestures. The sound and note gestures are illustrated in fig. 1. If the pointer enters the valid square from the left or right, the sound is gestured, and if it enters from the top, the note is gestured. If the pointer enters from below, the end-of-sequence gesture is made.

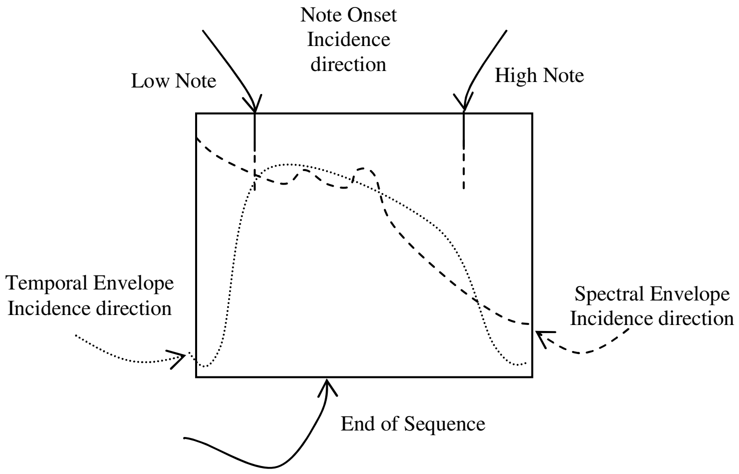

Note Onset Incidence direction
Low Note
High Note
Temporal Envelope Incidence direction
Spectral Envelope Incidence direction
End of Sequence

Fig. 1. The multiple musical gesture interface. A pointer is controlling either the identity of the sound, or the note sequence.

The brightness control function [3] (BCF) is rather similar to the timbre model, although significantly easier and cheaper to synthesize. The sound control, as illustrated in fig. 1 (left incidence direction), is valid for both synthesis methods. The irregularity is added by reversing the pointer direction during the sound gesture, as shown in fig. 2.

In the case of the timbre model and a skilled performer, the backward motion place affects the time or frequency location of the added irregularity, dependent on if the temporal or spectral envelope was gestured. Contrary to the timbre model control, the BCF irregularity control has no frequency location control. Instead, both shimmer and jitter are added on the time location corresponding to the distance in the direction of the x axis of the reversed gesture.

The note control is the same for the BCF as for the timbre model control [6]. A new note is initiated when the pointer is entering the valid square from the top. The

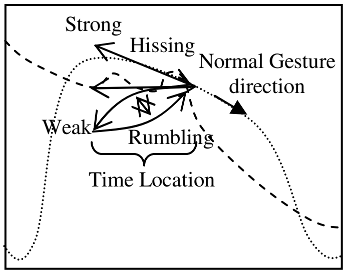


Fig. 2. Irregularity is added by reversing the sound gesture direction. Shimmer is affected when gesturing the temporal envelope, and jitter when gesturing the spectral envelope.

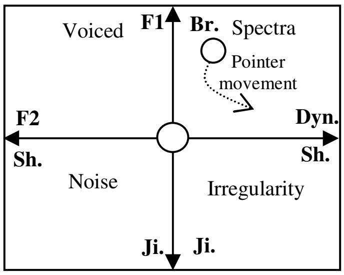


Fig. 3. The timbre gestures for the BCF

note is sounded as long as the pointer is inside the valid square. A vibrato is added, depending on the undulations in the x-axis direction during the downward motion.

The timbre gestures are made when the pointer enters and leaves inside the valid square, as illustrated in fig. 3.

The pointer has to enter inside the valid square, by ‘switching it on’. When this is done, it alters the parameters of the timbre graphs it enters. When the pointer exits, the value at the extinction point is retained. As an example, fig. 3 shows a timbre gesture in which the brightness is decreased, and made more dependent on the dynamics of the note played. The ‘timbre’ graphs are Spectra, in which the brightness value and dynamic sensitivity [9] are controlled, the irregularity and noise graphs, in which the low-frequency (rumbling) and high-frequency (hissing) irregularities are controlled, and the voice graph, in which a formant filter F1 and F2 frequencies are controlled.

3.2 Atomic Synthesis

Atomic noise [5] is in no way similar to a note-oriented synthesis. It has no pitch, no temporal envelope, and no spectral envelope. A recent extension, the perceptual atomic noise, changes the uniform frequency distribution and uniform amplitude values into a frequency distribution that is supposed to give approximately the same number of atoms for each critical band and amplitudes that give the same perceptual level for all frequencies, approximately inversing the filtering made by the auditory system, for a given theoretic listening level. Since this filtering is dependent on the intensity, the perceptual atomic noise dynamic is easily controlled.

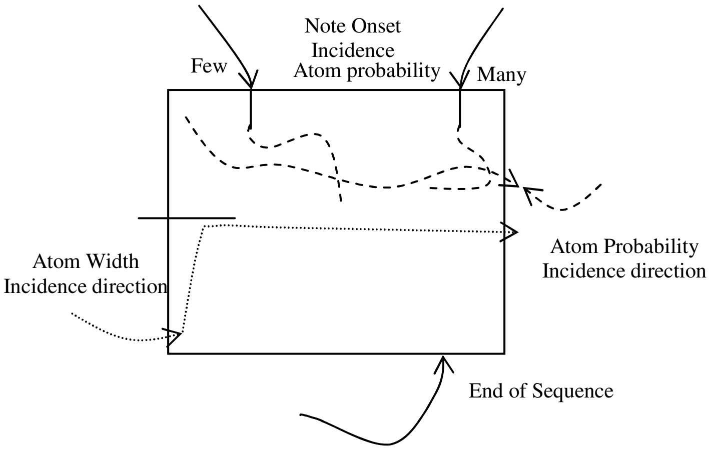


Fig. 4. The atomic noise model has the same note/sound control as the BCF. Pitch is replaced with atom probability, however.

The sound control of the atomic noise model has the atom width control to the left, and the atom probability to the right. While the note onset position decides the atom probability, this value is changed over time with the curve obtained form the atom probability incidence direction. The note dynamics controls the auditory inverse filter, which gives an effect similar to many musical instruments, with an increase in brightness for higher intensities. The timbre gestures affect a different set of parameters than the BCF parameters, as illustrated in fig. 5.

The timbre graphs of the atomic noise affects the listening level value and dynamic dependency, the periodicity weight and period of the time and frequency respectively

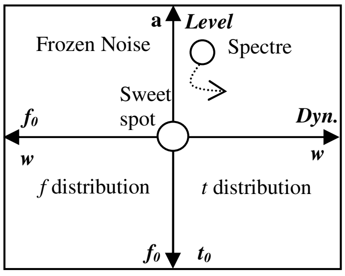


Fig. 5. Atomic noise timbre gesture possibilities

and the repetition rate and acceleration rate, in case the noise is repeated (frozen noise). In fig. 5, the listening level is decreased, but made more dependent on the dynamic level of the played note, thus making a less bright sound that becomes relatively more bright with the dynamic level.

4 Implementation

The goal of this work is to create a simple, intuitive and cost-effective interface for control over the synthesis and note sequences in a collaborative environment. The collaborative is understood here primarily in a traditional way, where each performer adds one sound/music, and the collaboration takes place in the creation of the resulting music, but it is also possible to have, for instance, one performer controlling the note sequence simultaneously to another controlling the sound parameters.

The multiple musical gesture interface has been shown to control different types of sound synthesis methods. The actual pointing device has yet to be defined, though. Ordinary computer pointing devices are not made to function independently and they also generally lack in freedom of extra-control movement, although they are efficient in navigation tasks [7]. One method to make a cheap interface that can control several synthesis methods at the same time is using a camera device. By pointing a laser pointer at and across the imagined valid square, and having a camera detecting this movement, a simple and cost-effective multiple musical gesture method is obtained. Although this approach does not give any force-feedback, it is believed that the combined auditory (music) and visual feedback remedies this. Finally, by using flashlights with different colors, several different synthesis methods or music sequences are controlled simultaneously, without additional cost.

The actual camera detection algorithm and synthesis are implemented in max/msp and jitter[1], using a standard usb camera. The unfortunate latency of common usb cameras is somehow ignorable by supposing that the actual valid square top border is situated higher than it is in reality. The multiple musical gestures is used in one interactive music piece to be performed at the CMMR conference in Pisa.

5 Conclusions

The multiple musical gestures interface contains the traditional note-onset metaphor, but permits additionally the in-detail shaping of the timbre of the musical sound, creation of musical sequences, and real-time control of many musically meaningful sound parameters.

Several sound models are presented, the timbre model, the harmonic brightness creation function, and the atomic noise. These sound models are chosen for timbre space is represents, the relatively cost-effective synthesis, the varied sound spectra produced and the possibility of real-time control of important timbre attributes. Each sound model is integrated into the multiple musical gesture interface by appropriate mapping of the sound parameters to the direction of incidence of the valid square of the interface. This allows the performer to create a musical sound by sound gestures,

[1] Max/msp and jitter are available from http://www.cycling74.com/

to produce a musical note by a note gesture, to produce a sequence of notes by a musical gesture, or modify the sound in real-time by continuous timbre gestures.

An example of a multiple musical gestures interface implementation is a color-sensitive camera that tracks the movement of a laser pointer in real-time. This enables the performer(s) to produce several musical streams simultaneously in the same interface, or to manipulate several aspects of one stream concurrently. This complicity is believed to enable a convivial musical activity.

References

1. Chowning J. M., The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. *Computer Music Journal* 1(2), 1977, 46-54.
2. Moorer, J. A. The synthesis of complex audio spectra by means of discrete summation formulas. *J. Audio. Eng. Soc.* 24, 9, Nov. 1976, 717-727.
3. Jensen, K., *Timbre Models of Musical Sounds*, PhD. Dissertation, DIKU Report 99/7, 1999.
4. Jensen, K., Envelope Model of Isolated Musical Sounds, *Proceedings of the DAFX*, Trondheim, Norway, 1999, 35-40.
5. Jensen, K., Atomic Noise, *Organised Sound*, accepted for publication, 2005.
6. Jensen, K., Multiple Musical Gestures in a 2D interface. *The 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*. Submitted, 2005.
7. Vertegaal R., Eaglestone B., A comparison of input devices in an ISEE direct timbre manipulation task. *Interacting with Computers*, 8, 1, 1996, 13-30.
8. Friberg, Generative rules for music performance: A formal description of a rule system. Computer Music Journal, Vol. 15, No. 2, summer 1991, 56-71.
9. Jensen, K., Musical Instruments Parametric Evolution, Proceedings of the ISMA, Mexico City, Mexico, np. (CDROM). 2002.
10. Jensen, K., *Irregularities, Noise and Random Fluctuations in Musical Sounds*. Journal and Music and Meaning 2, np. Spring 2004. http://www.musicandmeaning.net/.

Gran Cassa and the Adaptive Instrument Feed-Drum

Michelangelo Lupone and Lorenzo Seno

CRM – Centro Ricerche Musicali – Rome
http://www.crm-music.it
Conservatorio "A. Casella" – Dipartimento Musica e Nuove Tecnologie L'Aquila
http://www.mnt-aq.it

The physical-mathematical models of orchestral instruments represent an important theoretical and experimental support for the composer and for the application of new acoustic and performance criteria. Western music has linked its evolution to the transformation of instruments and performance techniques through the constant interaction between the expressive demands of the musical language (e.g. pitch range and control), acoustic requirements (e.g. sound irradiation level and type), sound emission techniques (e.g. ergonomics and excitation control). There is a constant interaction and reciprocal adaptation between the construction of the instrument and the composition and performance of music. For instance, consider how the tenth-century Viella evolved into the family of Renaissance Violas and then into the family of Violins. In terms of composition this coincides with the transition from monodic forms that duplicate voice and syllabic rhythm to the formal autonomy of instrumental music, with the spread of the frequency range, to the grand forms and orchestral ensembles of Baroque music. Executive technique is integrated in this process, since the player not only fills the role of agent producing the acoustic rendering, but also of expert demonstrating the criteria of agility and ergonomics of the instrument and inventing solutions of adaptation and virtuosity.

Considered in temporal terms, the transformations of musical language dating up to the last few decades, appear to us uncorrelated with the physical transformation of the orchestral instruments. The executive technique attempts to match the vibrational characteristics of the instrument with those of the language but, in many cases, results in aberrations of the physical system (e.g. multiple sounds of the winds, unconventional stimulation of the strings and resonant bodies) which make intensely complicated or aleatory the reproducibility of the acoustic phenomena and, as a consequence, also the notation and the prediction of the composer.

The use of electronics in processing instrumental sound has led to profound transformations, above all in the compositional and auditory terminologies, which immediately provide an answer to the expressive requirements of the musical language, but has favoured a real transformation only in a few instruments; by "transformation" is intended the extension or the characterization, both acoustic and executive (e.g. electric guitar). In serious music, where the power of transformation rests on the linguistic and technical system, electronics and the traditional instrument have for a long time been seeking interaction, integration and the sharing of sound development without however succeeding in losing their mutual identities. In many compositions electronics are used in parallel, it dialogues, integrates, draws out the instrument but does not change the instrument's acoustic and technical characteristics. In these compositions the presence of a new sound structure can be detected, but the action and control of the instrumentalist remain partial and are not necessarily a recognizable modulating cause of the sound. Above all when the musical passage utilizes processing of the instrument in real time - and in general when performing with live electronics - the main difficulty for the player is to render his technical and expressive style coherent with the resulting acoustic phenomena. The perceptive vicissitudes of the acoustic instrument and of electronics remain separate or uncorrelated; even when a rational reconstruction of the musical information is achieved during listening and the musical language supports the coherence of the information, we recognize the separation between the vibrational structure of the instrument, the process of electronic elaboration and the performance technique.

A study of the physical behavior of the instruments, its translation into mathematical models and, subsequently, its simulation with numeric calculation systems, appears to be the most feasible course to take for achieving specific forms of integration and the transformation or invention of new instruments coherent with present-day musical exigencies. Research in this direction investigates the complexities of physical reality for the purpose of constructing analytic and synthetic methods suitable for representing the phenomena involved. Obviously the aim is not to imitate the orchestral instruments for facile virtual utilization (typical limiting commercial trend), but rather to enlarge the knowledge of the vibrational phenomenon, to verify the models and to obtain acoustic confirmation of the logical and numerical process.

These were the premises on which the musical and scientific work was launched. This produced the composition *Gran Cassa* (Lupone, 1999), and subsequently led to the development of the instrument *Feed-drum*.

1 The Basic Instrument

The symphonic *bass drum*, the lowest-toned percussion instrument, was only added to the orchestra in the eighteenth century and by the next century had assumed the form we know today. In particular, the drum - which originally was narrow and long (as it still is in military bands) - was increased in size to 80-90 cm diameter and to 35-50 cm height of the shell with two heads of natural vellum fastened at the sides by systems which also regulated the tension. The biggest version, suitable for the largest orchestras, is the *imperial bass drum* with two heads of 102 cm diameter. It was on this type of instrument, lent by the L'Aquila Conservatorio and used for the first performance of *Gran Cassa*, that the preliminary experiments were carried out for both the work of musical composition and the project of the *Feed-drum* (Fig. 1).

The role of the bass drum - however essential and ever-present in the orchestra from Mozart onwards - is considered as secondary and limited to a few modes of sound emission: the drum roll (prolonged note), often finalizing crescendos and the reinforcement of the low tones of the orchestra in rhythmic sequences. Specific techniques were not studied, as in the case of the timpani, and typical strikers are the baton and the timpani sticks.

The idea of a musical work entirely based on this instrument originated from observation of the vibrational modes of the skin and from experiments made previously with the Planephones® and with the physical model of the string and the bow (Palumbi, Seno 1997).

Although the skin allows the excitation of a considerable number of high-frequency modes, their duration in time is not appreciable by the listener, apart from the timbric contribution to the attack phase of the sound. The possible variations of the mode of emission, adequate for a sufficient acoustic response of the resonator (shell), are limited and with scarce modulability. The basic frequency[1] , obtained by the tension of the skins, upper and lower, each bound to the edges with 16 mechanical tie rods, is influenced by the non-homogeneous distribution of the tensioning forces which contributes to render complex the spectrum of the real modes.

Fig. 1. Première of *Gran Cassa*, Alessandro Tomassetti plays an Imperial Bass Drum. Corpi del suono 1999 – Istituto Gramma, L'Aquila – Italy.

2 Experimental Work

Following a phase of listening and analysis of the sound characteristics of the instrument, adopting also unconventional modes of excitation such as rubbing and jetée of wire brushes, it was imperative for the composition of the musical work to

[1] We have preferred this term, here and subsequently, to the sometimes adopted "fundamental", since the latter should be more correctly reserved for the pitch frequency of tuned instruments.

identify and classify a wide range of possible sounds with different degrees of contiguity.

The experiments were made with the aim of achieving the following objectives:

1. variation of the basic frequency through the application of nodal constraints to the skin,
2. identification of timbres on the basis of type, mode and point of excitation,
3. sound modulation through glissandos, vibratos, portamento and rhythmic micro-articulation,
4. continuous and/or step variations of the dynamics, on the basis of the type of damping applied to the skin.

The characteristics of the traditional *bass drum* obviously do not permit the achievement of a range of acoustic results relevant to the proposed objectives. In order to explore the timbre richness of the attack phase and to isolate the vibrational modes, a system of electronic conditioning of the skin was created. Through the principle of feed-back, the signal produced by the excitation of the skin was returned to the skin itself in the form of acoustic pressure. The result was the infinite prolongation of the sound. The system controls the damping of the movement of the skin, and therefore the decay rate of the sound, and permits the isolation of high frequency modes by the combined action of the nodes present on the skin and of the amount of feed-back input energy. The stability of the signal obtained with this conditioning system made it possible to experiment and design on the skin surface a preliminary simplified map of the oscillatory modes based on the Bessel's functions. The map was limited to 13 diameters and 8 nodal circles (Fig. 2, Fig. 3), the latter divided into even semicircles (to the left) and odd semicircles (to the right).

Electronic conditioning of the instrument left the topology and primary acoustic features unaltered, but increased the scope of the vibrational criteria and control. This was used so that it was possible to distinguish the different pitches of various modes, to obtain the emission of long notes which could be modulated as those emitted by a stretched string and to adapt the acoustic energy independently of the emitted frequencies.

In order to maintain agility of execution and an adequate reproducibility of the phenomena, the first classification of sounds and performance techniques was limited to the use of fingers, hands and arms (Fig. 4). During the composition of *Gran Cassa*, experiments were also made with objects of different shapes and dimensions occupying wider or multiple nodal sections; this enabled us to increase further the sound possibilities, but the complexity of the vibrational phenomena involved an analysis also of the mechanical parts of the instrument in order to comprehend and reduce the dispersions as well as the non-linear contribution introduced by the vibrations of the structural materials and their combinations.

Given these complications, it was decided to plan and realize a new instrument, the *Feed-drum* (Fig. 5), for the purpose of not only extending the acoustic possibilities, but also of permitting the ergonomic use of new executive techniques. In particular, the vibrational attitude was transformed by eliminating the lower skin, a decision

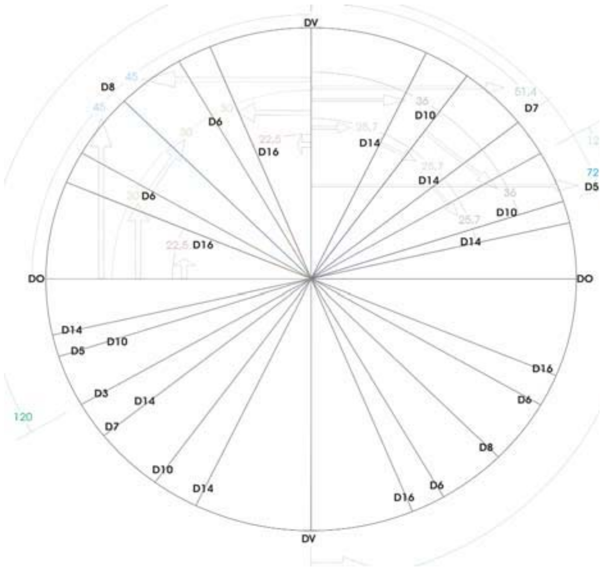


Fig. 2. Map of the first 13 nodal diameters

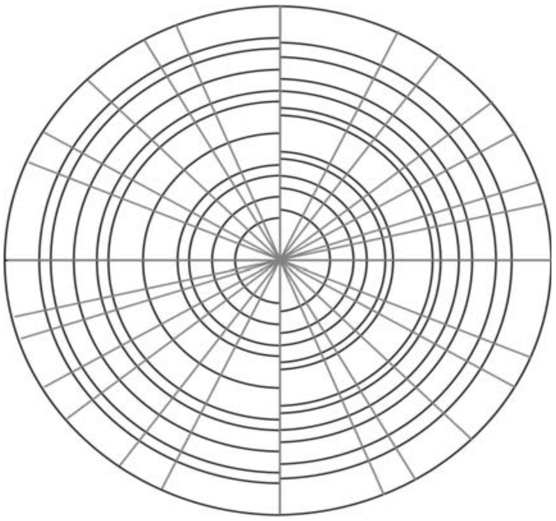


Fig. 3. *Feed-drum*, first map with 13 diameters and 8 nodal circles

which simplified the tuning of the instrument's basic frequency (30 Hz) and reduced the excitation rise time in the upper modes. A synthetic membrane was applied with isotropic characteristics and high flexibility on which the previously described map was drawn, with colors that made the areas of performance more visible. The shell and the tensioning hoop were realized in steel and aluminium; in particular, the

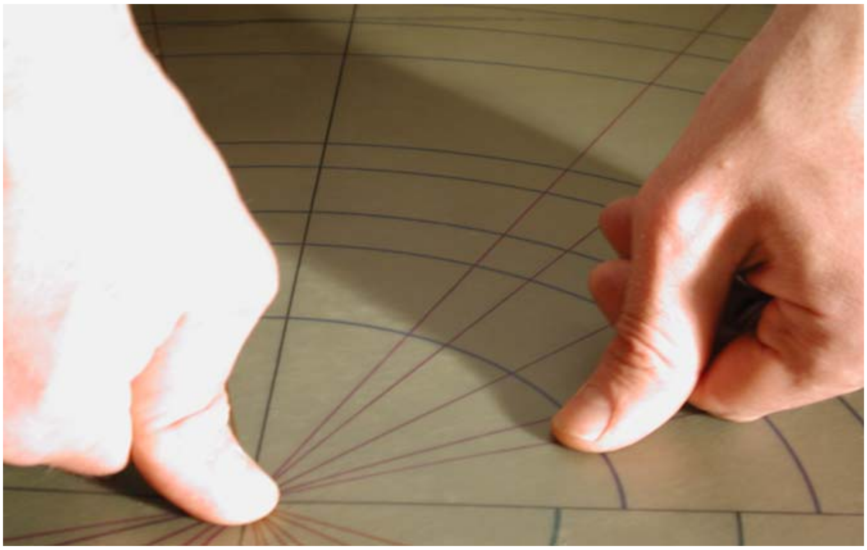


Fig. 4. *Feed-drum*, one of the performance techniques for the excitation of high frequency modes

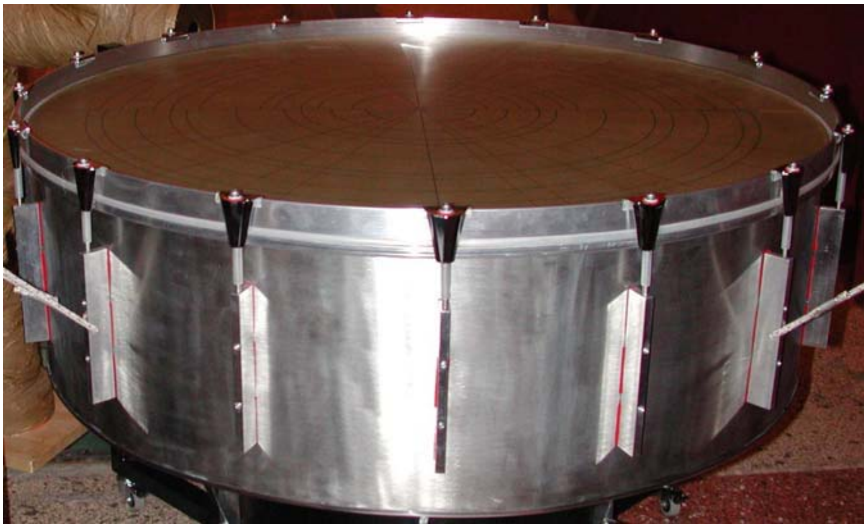


Fig. 5. *Feed-drum*

tensioning hoop was made stiffer while the height was reduced and the adhesion surface increased. The suspension system was realized in such a way as to separate the *Feed-drum* completely from the supporting structure on the ground; all the mechanical parts, which were in contact with one another, were separated by an intermediate layer of antivibrational material.

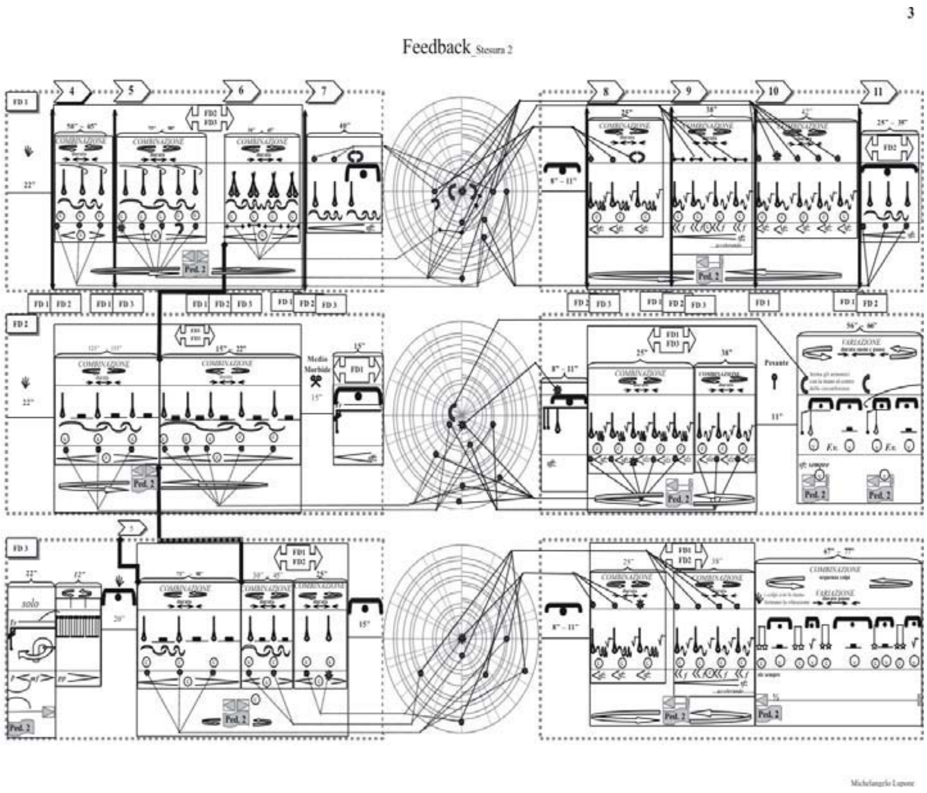


Fig. 6. *Feed-back – 3 Feed-drums* (2002), excerpt from the score

Despite the fact that there were still many aspects to be studied, it was possible to verify on the *Feed-drum* the reproducibility of the classified sounds and of the modulations, the adroitness of the excitation and control modes, the extension in frequency and the pitch characteristics. This facilitated drafting the performance score of the composition *Gran Cassa* and subsequently of the composition *Feedback* (for 3 *Feed-drums*) (Fig. 6) where there were, in addition to the usual indications of rhythmic practice, the forms and points of excitation of the membrane, the quantity of feedback input energy, the frequency and duration of the sounds, the point intensity, the types of modulation (vibratos, glissandos, portamenti), the range and velocity of the modulating action.

3 Theory of Operation – A Draft

The behaviour of the *Feed-drum* is extremely complex and many of its aspects still have to be clarified. We will attempt to illustrate here the known elements, those conjectural and those remaining to be defined.

The oscillation modes of a circular non-rigid membrane, pegged down and stretched along its rim, are known from literature. In a conservative model (that is, without dissipations and acoustic irradiations and therefore "in vacuum"), the oscillation modes of a membrane of radius a have the form in cylindrical coordinates

$$z(\rho,\varphi,t) = R(\rho)\cdot\Phi(\varphi)\cdot\cos(\omega\cdot t) \tag{1}$$

where: $R(\rho) = Jn(m,k\rho)$ and $\Phi(\varphi) = A\cdot\cos\left(m\cdot\varphi+\varphi^0\right)$ and where $Jn(m,x)$ are Bessel functions of the first kind and of an order m. φ^0 is an arbitrary phase dependent on the initial conditions (there cannot be any privileged directions, since the problem applies to a circular symmetry).

Owing to the constraint on the rim, $R(a) = Jn(m,k\cdot a) = 0$, where a is the radius of the membrane; this allows calculating k (wave number) which is discrete and dependent on two indices (m,n): $k_{m,n} = R_{m,n}/a$, where $R_{m,n}$ is the nth root of the Bessel function of order m, $Jn(m,x)$.

Hence (1) becomes $z_{m,n}(\rho,\varphi,t) = A_{m,n}\cdot Jn(m,k_{m,n}\rho)\cdot\cos\left(m\cdot\varphi+\varphi^0_{m,n}\right)$.

Determination of the wave number is therefore possible by determining the roots of the Bessel function of the first kind. Once the roots and wave numbers are determined, the angular frequencies peculiar to the modes are given by: $\omega_{m,n} = k_{m,n}\cdot c$ where c is the velocity propagation of transverse waves in the membrane, $c = \sqrt{T/\sigma}$ where T is the stretching force of the rim and σ is the surface density of the membrane. However, c can easily be estimated on the basis of

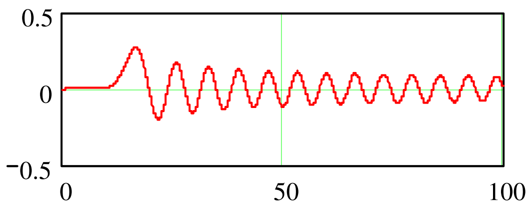


Fig. 7. *Jn(*1,*x)*

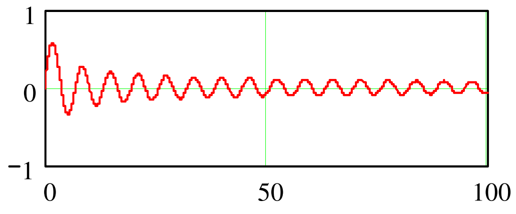


Fig. 8. *Jn(*15,*x)*

the frequency ν_1 of mode $(0,1)$, the lowest of all (basic frequency), taking into account that $R_{0,1} = 2.405$:

$$c = \frac{2 \cdot \pi \cdot \nu_1}{R_{0,1}} \cdot a$$

For the *Feed-drum*, $a = 0.51m.$ and $\nu_1 = 30Hz$, and therefore $c = 40 \cdot m/sec$.

Irrespective of the order of the Bessel functions, the root base tends to π for $m \longrightarrow \infty$ [2]; in addition, Bessel functions of different order do not have coincident roots (an important consideration for the purpose of the *Feed-drum*).

The exact calculation of the roots can only be realized numerically, a task which is not particularly difficult given the oscillatory character (even if not periodical) of Bessel functions. In fact, the roots of these functions are each comprised between a maximum and a minimum or vice versa.

Calculations of the frequencies for the modes up to 5 octaves above the "basic frequency" (960 Hz for the *Feed-drum*) gives the following distributions of frequencies and modal density:

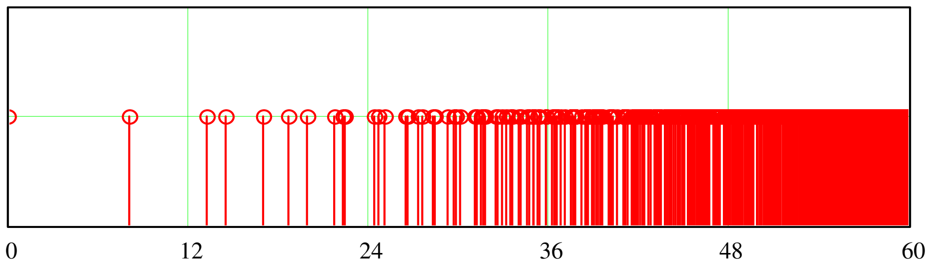


Fig. 9. Modal frequencies in semitones with respect to the basic frequency

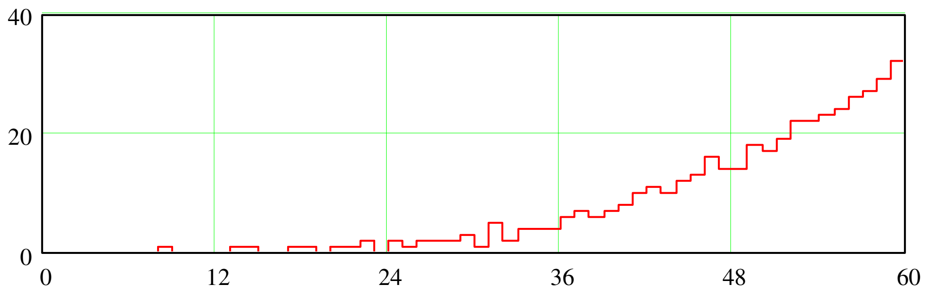
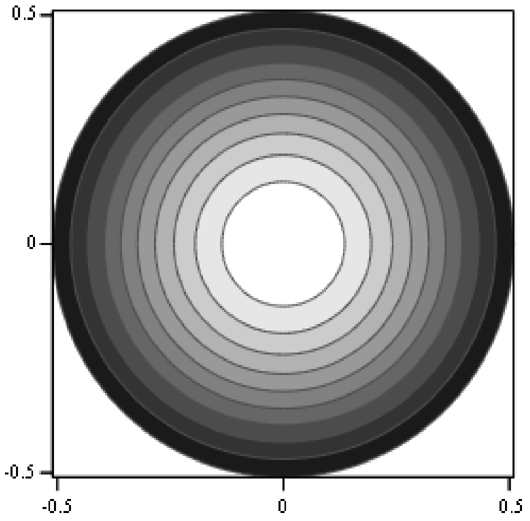
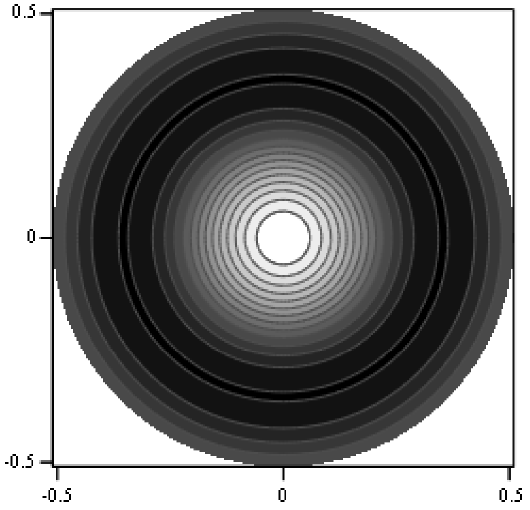


Fig. 10. Modal densities: the number of modes per semitone in the ordinate; in the abscissa the modal frequencies in semitones with respect to the basic frequency

The index *m* is responsible for the creation of nodal diameters, the index *n* for that of nodal circles. In general, the pattern of the modes is simply correlated to the indices, as can be seen from the diagrams given below.



(0,1) $v = 30Hz$ (basic frequency)

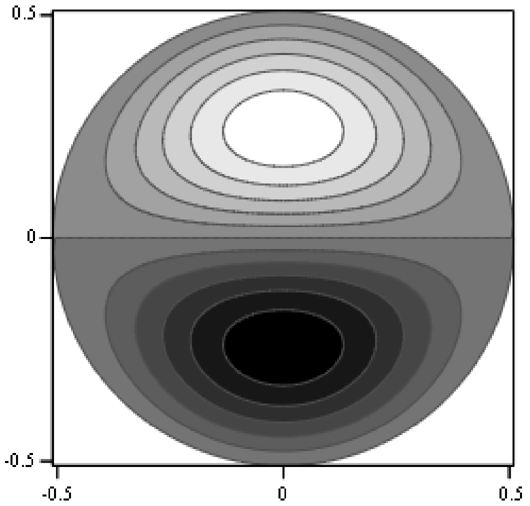


(0.2) $v = 68.9Hz$

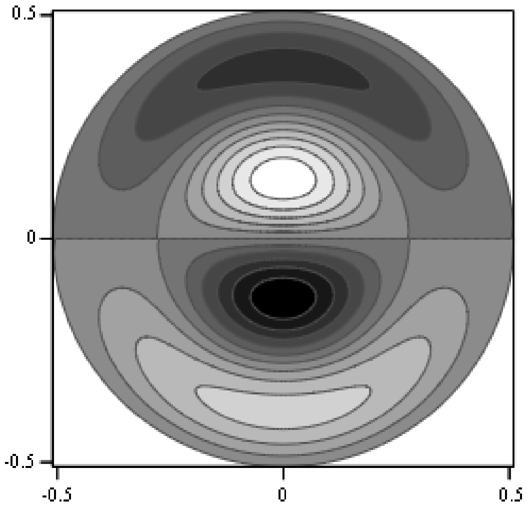
Fig. 11. Modal maps

4 Conditioning System and Implementation

Excitation of the membrane is via a loudspeaker (Ø = 45 cm.) and a 11 cm-long wave guide (designed to convey maximum acoustic pressure between the center and 1/3 of the radius); that is, fairly short as far as the form factor is concerned. It proved



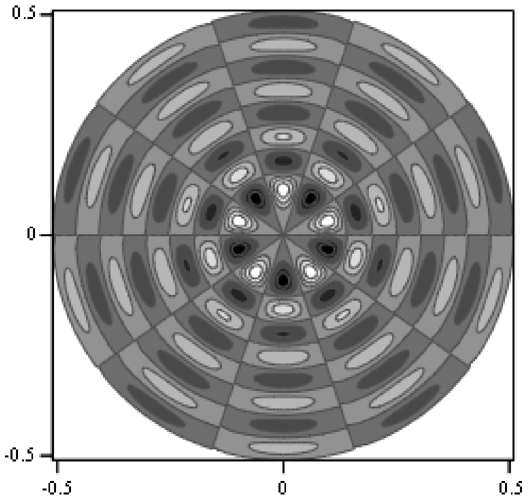
(1,1) $\nu = 47.8 Hz$



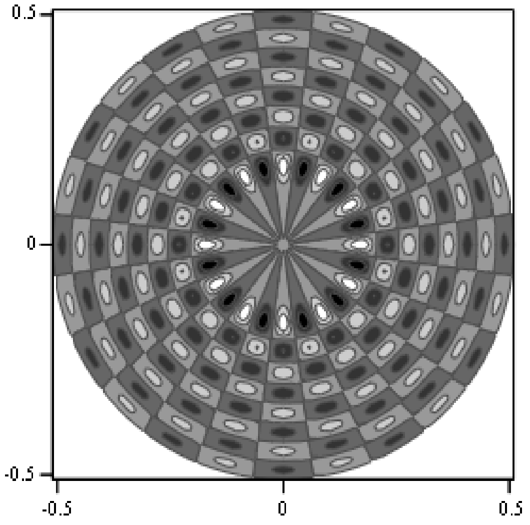
(1,2) $\nu = 87.5 Hz$

Fig. 11. (*Continued*)

fairly easy to obtain, in addition to the 30 Hz basic frequency, the 68.9 Hz frequency corresponding to the mode (0,2). It was on the contrary impossible to obtain the frequency of 47.8 Hz corresponding to the mode (1,1). At these frequencies, the behavior of the air excited by the loudspeaker can presumably be schematized with a piston motion, which exercises an almost uniform pressure on the membrane. A uniform excitation is poorly compatible with the modal form (1,1).



(5,8) $\nu = 396.8Hz$



(12,8) $\nu = 517Hz$

Fig. 11. (*Continued*)

The loudspeaker was driven by an electric power signal, generated by a feed-back system that sampled the signal issued by a piezoceramic sensor placed on the rim and detecting deflection of the membrane. In this way a "multimodal" oscillator was obtained generating a feed-back on a resonant element, the membrane. The loop gain was controllable by a pedal.

5 Intonation of the Upper Modes

Intonation is through the combined action of the negative feed-back gain and of the pressure on one or two points of a nodal line. The effect of the pressure can be schematized in a first approximation as dual: on one hand the introduction of a constraint on the pressure points, on the other a shift of the “work point” of the membrane around a slightly higher tension, and therefore an increase of the transverse wave velocity c . Consequently, all the frequencies move upwards. It is a matter of a shift mechanism, a “pitch-shift”, in the sense that the frequencies of the modes are all multiplied by a common factor, therefore leaving unchanged their relationships. In point of fact this effect has been encountered in practice and is utilized for obtaining the vibrato. The term “pitch-shift” is however improper in this case, since the spectrum of the partial tones of the membrane is not harmonic and as a result a pitch is not definable[2].

The apposition of constraint points ($z = 0$) has the effect, as a rule, of inhibiting every mode which has no set of nodal lines passing through all the aforesaid points, not even with an opportune choice of φ^0 .

For example, pressing the center of the membrane makes all the modes with m = 0 becoming impracticable, since this point is invariably an antinode for these modes. Pressure on any other point of the membrane (speaking theoretically) makes all the modes with $m \geq 1$ practicable, since it will always be possible to have a nodal diameter passing through that point. In practice, since the constraint is not perfect, preference will be given to the mode which possesses both a nodal diameter and a nodal circle passing through that point. The consequence of the fact that the functions of Bessel have no coincident roots is that the modes of different m order cannot have coincident nodal circles. Even modes with the same m and different n obviously cannot have coincident nodal circles. Two different modes can, on the other hand, have coincident nodal diameters if the ratio of their indices m is an integer number. A single pressure point different from the centre identifies therefore a mode only having a diameter and a circle passing through that point. The points which “discriminate” better the frequency modes are, however, those near the centre, because the nodal circles become densely-packed towards the perimeter and a single point therefore tends to have many of them very near to it. Consequently, it is the first nodal circle, the innermost one, which best discriminates the modes, as is also shown by a variance analysis.

In theory, pressure on any two points of the membrane could create constraints incompatible with any mode.

However, all these considerations are better limited to modes of relatively low order. In fact it can be presumed that the approximation of the non-rigid membrane results less valid with the increase of the mode order, since the node base tends to become comparable with the thickness of the membrane itself.

[2] It is known that the timpani (kettledrums) are endowed with pitch, but this is obtained thanks to the kettle and to the interaction with a great mass of air. See [1] for a more in-depth discussion.

There are also other considerations. The classic equation of the membrane generally used for obtaining the modes (see 3.7, p. 69, of [1]) is, as already mentioned in this text, entirely conservative and does not take into account either dissipation due to internal friction or irradiation. The latter both being mechanisms that dampen the partials, causing their decay in the absence of an exciting force.

A symbolical solution of the equation corresponding to the vibro-acoustic movement described is definitely impossible, even if greatly simplifying hypotheses are adopted. It is certainly possible to solve it with numerical methods (such as FEM, BEM, etc.) but even in this case, if acoustic-elastic coupling and internal dissipations of the membrane are to be taken into account, the problem still remains extremely delicate and the results should be subjected to thorough experimental verification.

However, even in the absence of a solution, it is possible to note that decay of the partials is in any case connected with the merit factor (Q) of their resonance and provokes a widening of the spectral line, always more marked the more the relative mode is damped. The internal frictions are proportional to the velocity of variation of the local curvature, which increases with the frequency. It can therefore be presumed that, similarly to stretched strings, damping of the modes increases with their frequency. Consequently, in the upper spectral areas, where the modes are close together and massed (see Figs. 9 & 10), the transfer function of the membrane is more continuous than discrete, with moderate peaks on the modal frequencies. In these areas the modes which can be excited are less precisely definable and depend on the loop gain and on the frequency characteristics of the negative feedback electronic circuit. Conversely, the passage from one mode to another of adjacent frequency has little influence on the resultant frequency.

Construction of a future improved map for excitation of modes must therefore foresee a judicious choice of the pairs of points which offer the most significant discrimination between modes.

In addition, the modal frequencies should be verified experimentally, since it can be presumed that the frequencies of some modes deviate from their nominal values owing to the presence of the actuator with relative wave guide which has a beam width equal to 1/3 of the membrane diameter. The measurement of these deviations cannot be determined reliably with theoretical considerations, since the overall model is too complex and can only be solved (as already shown) with numerical methods.

6 Conclusions

The trials carried out to date with composers and percussionists, both classical and jazz, were received with an enthusiasm which stimulated suggestions for extending the control criteria, the use of special strikers of various forms and dimensions, and the application of independent hand techniques. No need was found for substantial structural modifications to the *Feed-drum*. Subsequent developments will principally concern ergonomic aspects, with the compilation of more precise nodal maps, probably of different conception, and of simpler and immediate use. In addition, improvements are quite conceivable in the electronic conditioning system and in its operation for the purpose of improving controllability of the emission of high overtones. These revisions of the project will be preceded by a series of measurements with the object of defining more thoroughly the behavior of the *Feed-drum*, both with the loop open or closed, through assessment of the transfer functions and exact measurement of at least one significant subset of modal frequencies and their comparison with theoretical predictions.

References

[1] M. Lupone, Corpi del suono, Istituto Gramma – L'Aquila, 1999
[2] M. Palumbi, L. Seno, Physical Modeling by Directly Solving Wave PDE, In Proc. of the 1999 International Computer Music Conference, ICMA 1999
[3] N.H. Fletcher, T.D. Rossing, The Physics of Musical Instruments, Springer, 1991
[4] S. Finch, Bessel Functions Zeroes, INRIA, October 2003 http://pauillac.inria.fr/algo/csolve/bs.pdf

Generating and Modifying Melody Using Editable Noise Function

Yong-Woo Jeon, In-Kwon Lee, and Jong-Chul Yoon

Dept. of Computer Science, Yonsei University,
Seoul, Korea
{ywjeon, iklee, media19}@cs.yonsei.ac.kr
http://visualcomputing.yonsei.ac.kr

Abstract. This paper introduces a way to generate or modify a melody using the editable noise function. The band-limited random numbers generated by the noise function are converted to the various property values of notes such as pitch and duration. Using this technique, we can modify an existing melody to produce new, similar melodies. The noise values can be edited, if necessary, while preserving the statistical characteristics of the noise function. By using this noise editing method, the noise function can generate a melody that satisfies given constraints.

1 Introduction

Composing music with mathematical concepts or formal processes has been studied since the early twentieth century. Many composers have produced works in this way. The stochastic music introduced by Xenakis [1, 2] and the use of pure randomness in composing and performing music by John Cage [3, 4] are typical examples of it.

In this paper, we consider Perlin noise [5, 6] as another tool that gives accidental elements to music. Perlin noise, which has been used in many applications in computer graphics [6], is different from white noise (plain random numbers). It is a function that satisfies the conditions for the ideal noise function, which is bounded, band-limited, non-periodic, stationary, and isotropic [6]. Because of its properties, the Perlin noise function is also proper for giving controlled randomness to music.

Music can be parameterized with many property values such as pitch, duration, etc. This paper explains how the noise function generates or modifies melodies by applying noise values to the parameters of the melodies.

Although various probabilistic and algorithmic composition methods, such Markov chain and cellular automata [7, 8, 9] have been proposed, it is hard to implement intuitive composing and modifying processes that include randomness but are controllable and predictable. With the editing power of the noise function, we can give some user-defined constraints to the generated music, which permits users to control the composition process. For example, when we modify an existing melody, constraints can be given to prevent some notes from being changed while other notes are changed by noise values. The noise editing method preserves the statistical characteristics of the noise function, which keeps the fair randomness of the output while satisfying the constraints defined by users.

2 Noise Function and Editing

Our method is based on Perlin's gradient noise algorithm. The following outline explains how the noise generation algorithm operates when a 1D coordinate x is supplied as an input [5] (see Fig. 1(a) for an example of 1D noise).

Step 1: Generate M PRNs (pseudorandom numbers) in $[-1, 1]$ and store them in a table $G[0 : M-1]$ of size M. Prepare another table $P[0 : M-1]$ that holds a random permutation of the set of integers from 0 to $M-1$.
Step 2: Obtain the integer interval $[q_0, q_1]$, where $q_0 = \lfloor x \rfloor \bmod M$, and $q_1 = (q_0 + 1) \bmod M$.
Step 3: Obtain two PRNs $g_j = G[P[q_j]]$, $j = 0, 1$.
Step 4: The noise value is computed by: $\text{Noise}(x) = (1-s(d_0))g_0 d_0 + s(d_0) g_1 d_1$, where $d_j = q_j - x$, $j = 0, 1$, and $s(t) = 6t^5 - 15t^4 + 10t^3$ is an ease curve. Note that $-1 \leq \text{Noise}(x) \leq 1$.

Taking sum of noise functions of various frequencies is more suitable in many applications. This kind of fractal sum [10] has been used for modeling more complex change in various frequencies. The general fractal sum of N noise functions having different frequencies f_j and amplitudes a_j is defined as $\text{Fsum}(x) = \sum_{j=1}^{N} a_j \text{Noise}\,(f_j x)$, where $a_j \geq a_{j+1}$ and $f_j \leq f_{j+1}$, for $j = 1, \ldots, N-1$. Fig. 1(b) shows an example of a fractal sum.

Yoon et al. [10] introduced the noise editing method to generate a PRN table G satisfying a given user constraint: $\text{Noise}(x_*) = H_*$ which represents a user's demand to fix a noise value at a specific position x_* to a desirable value H_*. To preserve the statistical characteristics of the noise function, they suggested an optimization-based method to minimize the Chi-square statistic of the edited random number distributions. The Chi-square statistic is used to test whether an unknown distribution is near a desirable distribution. In noise editing, the Chi-square statistic is minimized to make the new distribution as similar as possible to the uniform distribution (the pure random distribution). Thus, the minimization problem is stated as follows:

$$\text{Minimize} \;\; D^2 \;\; \text{subject to} \;\; \text{Noise}(x_*) = H_*$$

where D^2 is a Chi-square statistic of the target distribution of the table G. Fig. 1(c) shows an example of the noise editing result using this method. We can see that the edited result preserves the statistical characteristics of the original noise function, keeping the function shape similar to the original function. However, a number of modifications of the noise function without considering the statistical characteristics make the resulting function totally different from the original one (see Fig. 1(d)). Refer to [10] for more details about noise editing.

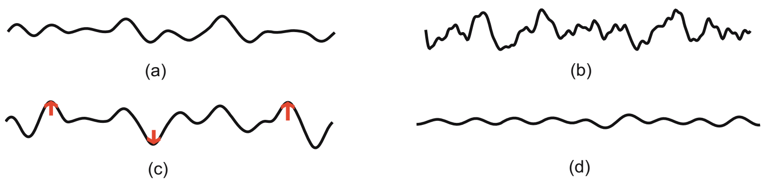


Fig. 1. (a) 1D noise function, (b) fractal sum, (c) a result after several editing operations of the original noise function in (a) (arrows denote the edited positions), (d) a modified function without considering the statistical characteristics of the noise function

3 Applying Noise to Melody

Noise values can be applied to various properties of melody notes such as pitch, duration, note-on time, volume, tempo, etc.

Noise to Pitch. Pitch is one of the most important properties of a note, and a relatively small change is recognized easily. In this work, a pitch is quantified as a MIDI note number, and the noise function is applied to the pitch value to modify the pitches of an existing melody or generate a pitch sequence for a new melody.

Noise values–Noise(t)s–can be added to the pitch values–Pitch(t)s–of the notes in an existing melody to generate a new melody, where t is the note-on time of the note:

$$\text{Pitch}'(t) = \text{Pitch}(t) + s \cdot \text{Noise}(t), \tag{1}$$

where s is a scale factor. Or, we can use the ordered index n of each note as a 1D variable:

$$\text{Pitch}'(n) = \text{Pitch}(n) + s \cdot \text{Noise}(n). \tag{2}$$

The resulting melody gives listeners feelings similar to the original melody, but it is not exactly the same as the original music. Fig. 2(b) shows a modified version of an original melody in Fig. 2(a).

To make a note at t_* have a specific pitch H_*, a constraint $\text{Pitch}'(t_*) = H_*$ is set, and the noise function is edited to satisfy the constraint. Fig. 2(c) is another version of a modified melody that has some constrained-pitch notes represented with the arrows.

A completely new melody can be created by noise function. Given a skeleton note or set of notes, the noise values are added to create a unique melody line:

$$\begin{aligned} \text{Pitch}'(t) &= \text{Pitch}_{skeleton}(t) + s \cdot \text{Noise}(t) \quad \text{or} \\ \text{Pitch}'(t) &= \text{Pitch}_{skeleton}(t) + s \cdot \text{Fsum}(t). \end{aligned} \tag{3}$$

Fig. 3 shows an example of a melody line created by adding the fractal sum values to the middle C notes.

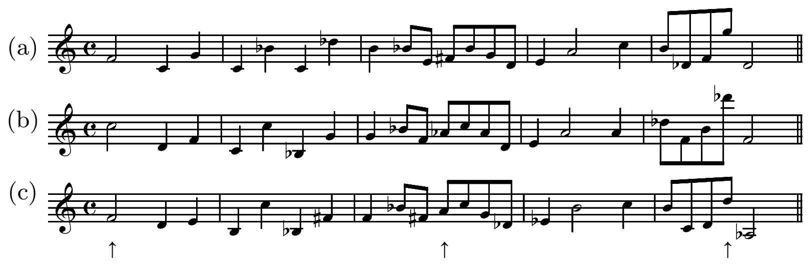


Fig. 2. Example of modifications of pitches in a melody: (a) original melody, (b) modified melody, and (c) modification with noise editing technique: one note (F) is constrained to be fixed as in the original melody, and the changes of two pitches (A and D) are explicitly given as the constraints

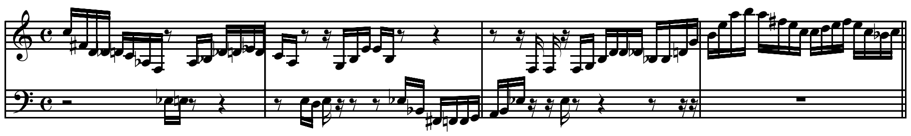


Fig. 3. Example of generating melody: pitch sequence is generated by the fractal sum of noises: Pitch$(t) = 72 + 12\text{Noise}(t) + 6\text{Noise}(4t) + 3\text{Noise}(16t)$

Noise to Timing Information. Note-on times and durations are perturbed using noise function, too. For applying noise, duration and note-on time values are quantified as tick values of the MIDI protocol. In this case, the resulting notes may be out of their original measures and beats, which should be corrected in a postprocessing stage. The total length of the melody may also be changed, but because the noise function is unbiased and the mean value of the distribution is almost 0, the change is usually very little.

Noise to Other Properties. Noise can be applied to many other properties of a note such as dynamic, tempo, etc., to produce other effects as follows:

- Humanizing a melody by perturbing tempo and the dynamics of notes.
- Simulating *crescendo* and *decrescendo* by modifying the dynamics of notes with a noise value per a measure or phrase.
- Simulating *accelerando* and *ritardando* by modifying the tempo of a melody with a noise value per a measure or phrase.

4 Conclusion and Future Work

Noise function makes the modification of an existing melody easy and fast. The modified melody shares the basic form of the original melody, but the details are randomly different. The extreme changes that can be caused by white noise are not found in the modification because of the ideal band-limited feature of the noise function. By noise editing, it is possible to control the composition or modification process precisely to reflect a composer's intent in the work.

There are some limitations to this approach. When pitches are generated using the noise function, many atonal notes can occur, so that the original tonal melody loses its tonality. If note-on values are changed, beat feel is decreased, too. Applying noise is more suitable for atonal, non-beat, and avant-garde music based on probability and accidents than classical music structured by tonality and meters. That is, the noise method can be a tool for algorithmic composition with randomness and some constraints from a composer that should be satisfied. To produce more practical output, we need to postprocess the output to revise the atonal notes to be tonal using other musical rules. For example, we can slightly shift an atonal note up or down to the nearest tonal note.

We are trying to find more properties or elements of music to which the noise value can be applied. For example, editing a fine tune or micro control of dynamics would generate natural effects such as *vibrato*, *tremolo*, and *glissando*. Timbres and spatial parameters can also be considered as targets of the research. Though we considered only 1D noise in this paper, the noise function can be extended in any higher dimension. It will be interesting to construct more complex musical structures in higher dimensions at which the noise function can be applied.

Acknowledgement. This work was supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

References

1. Harley, J.: The Electroacoustic Music of Iannis Xenakis. Computer Music Journal 26:1 (2002) 33-57
2. Pape, G.: Iannis Xenakis and the "Real" of Musical Composition. Computer Music Journal 26:1 (2002) 16–21
3. Pritchett, J., Whittall, A.: The music of John Cage (Music in the Twentieth Century). Cambridge University Press (1996)
4. Kostka, S.: Materials and Techniques of Twentieth-Century Music. Prentice Hall (1999)
5. Perlin, K.: Improving noise. SIGGRAPH '02 Proceedings (2002) 729–735
6. Ebert, D., Musgrave, F.K., Peachey D., Perlin, K., Worley, S., Mark, B., Hart, J.: Texture & Modeling: A Procedual Approach 3rd Edition. Morgan Kaufmann (2002)
7. McAlpine: Making music with algorithms. Computer Music Journal 23:2 (1999) 19–30
8. Millen, D.: An Interactive Cellular Automata Music Application in Cocoa. Proceedings of ICMC (2004)
9. Miranda, E.R.: Composing Music with computers. Focal Press (2001)
10. Yoon, J-. C., Lee., I-. K., Choi, J-.J.: Editing Noise. Journal of Computer Animation and Virtual Worlds 15:3 (2004) 277–287

Unifying Performer and Accompaniment

Lars Graugaard

Aalborg University Esbjerg,
Department of Software and Media Technology,
Niels Bohrs Vej 6,
DK-6700 Esbjerg, Denmark
`lag@cs.aaue.dk`

Abstract. A unique real time system for correlating a vocal, musical performance to an electronic accompaniment is presented. The system has been implemented and tested extensively in performance in the author's opera 'La Quintrala', and experience with its use in practice is presented. Furthermore, the system's functionality is outlined, it is put into current research perspective, and its possibilities for further development and other usages is discussed. The system correlates voice analysis to an underlying chord structure, stored in computer memory. This chord structure defines the primary supportive pitches, and links the notated and electronic score together, addressing the needs of the singer for tonal 'indicators' at any given moment. A computer-generated note is initiated by a combination of the singer – by the onset of a note, or by some element in the continuous spectrum of the singing – and the computer through an accompaniment algorithm. The evolution of this relationship between singer and computer is predefined in the application according to the structural intentions of the score, and is affected by the musical and expressive efforts of the singer. The combination of singer and computer influencing the execution of the accompaniment creates a dynamic, musical interplay between singer and computer, and is a very fertile musical area for a composer's combined computer programming and score writing.

1 Introduction

Alignment of score and electronics in real time became a research area in 1984 when Barry Vercoe and Roger Dannenberg independently presented their seminal papers on this concept [33, 9, 10]. Much work has been done since then [7, 27], and recent new directions have been in combination with the much larger research area of affective computing, such as correlating intended emotion to musical analysis in real time emotion tracking [13, 17].

In this paper a practical implementation of a new concept of correlating a performer to an accompanying electronic soundscape is presented, where timbre and expressivity of a performer is analyzed for subsequent use in sound synthesis. The system addressed practical needs in an artistic project. Proven and reliable algorithms and concepts of data handling from previous research was used upon due scrutiny, such as spectral peak estimation [30, 20] and high-level pitch manipulation as sets of pitch-classes [11].

It was not the intention to build a new system from bottom up, and the scientific contribution of this paper reside in describing the first implementation of a generalized system for sonically unifying a performer in real time with an electronic soundscape intended to support and cue the performer, while at the same time enhancing the performer's affective presence in the accompaniment. Specifically,

– The paper is a contribution to music intelligence as a subset of machine intelligence, and it is shown that a computer programme flexibly can adapt a soundscape to a performer in real time.
– The affective content of a musical performance can be used to enhance sound synthesis, without necessarily imprinting the spectral content of the performance analysis.
– The system is based on a structure whereby a performance and its accompanying electronic soundscape can correlate with a predefined informal chordal structure, for instance notated in a performance score.

The report of the artistic project that gave rise to the system is encompassed as one example of its use, and it is seen that the system can be used as a general tool when coordination of an underlying chordal structure in a musical score with a performer's expressivity is called for.

The artistic contributions are best appreciated by listening to excerpts of the performances at www.graugaard-music.dk/pages/la_quintrala_ex.html. Musical quality is of a highly subjective nature, but the system presented in this paper can be considered successful in regards with producing an accompaniment that has proved its stated purpose through many performances.

2 Background

In 2002 I was commissioned to compose an opera for five singers and interactive computer as a joint Danish, Swedish, and German commission through EU's Culture 2000 Foundation. It was to be scored for five singers and interactive, computer generated music, and it was to have a duration of two hours. The opera was later entitled 'La Quintrala' after its main character [15], and it was premiered in Copenhagen on September 2nd 2005 at Den Anden Opera. A subsequent 28 performances took place in Sweden and Germany, with the last performance in Sweden on November 11th 2005.

While composing the vocal parts, it became clear to me that a versatile and flexible tool for unifying the electronic accompaniment with the vocal lines and for supporting tonal structure was needed. Specifically, it was necessary to tonally bind the singers to their accompaniment by cueing their pitches and providing supportive harmonic structures, to sonically unify their voices with the electronics, and to provide sufficient compositional freedom for serving the dramatic and musical needs of the composition. No existing software tools seemed to serve my purpose.

3 Functionality Requirements

Opera is a musical drama where feelings of love, longing, hope, fear, and hatred are the essence of tragedy and drama. The electronic score for ‘La Quintrala’ would have to be able to project emotions into the soundscape as they would be exposed by the singers on stage. But the soundscape would also have to be able to relate the pitches of a singer to a chord succession to be synthesized, thereby providing the singers with tonal orientation. It is hard not to see these two requirements as opposites: the needs of a singer at a given point can be very different from the musical needs according to the composer. This, however, is not a characteristic of interactive music as such, yet attempting to meet these two requirements could turn the soundscape into an effective, compositional tool for shaping the formal development of the dramatic content.

Questions of timing and event coincidence needed to be decided on at an early stage as it would influence the system design. My interactive music generate exactly timed events ‘on the fly’ by letting the performance analysis affect the chosen synthesis method(s). I therefore didn’t need an accompaniment tool that would emphasize exact timing of events or action advance based on timing prediction, but I would need as low a latency as possible to get good time coincidence. This is fundamentally a compositional issue, to have the performer define the perceived rhythm periodicity. When the rhythm texture is sufficiently complex to be ambiguous in itself, yet sufficiently simple to maintain the feeling of pulse, then the performer would ‘define’ the periodicity through the periodicity of the sung melismas. So, rather than classical score following, what I needed can be termed score correlation.

4 Extracting Expressive Data

The analysis of the voice could have been done with pitch estimation only, as this would have given information of the fundamental of the note sung. This representation would be directly applicable to the score notation of the voice and the underlying chord structure. But pitch estimation would not contain any expressive data, since this is embedded in the spectrum of the sound, and is discarded in the process of pitch estimation. A musical score is, on the same token, very limited with respect to the actual sound of the music notated, that is, the auditory information that arrives at the listener’s ears [32].

The analysis had to provide an indication of the emotional triggers which the singers would embed in their vocal performance. Expressive information manifests itself in fluctuations of spectrum components at the onset of and during a note, in changes of dynamics, and in spectral and amplitude contours. Expressive information is readily appreciated by expert and non-expert listeners alike, as the continuous, complex sound reaches their ears. But extracting such expressive content of an audio signal is not easily done. In fact, algorithms created for this purpose are in some respects still easily outperformed by even non-expert music listeners as described in [26]. Fortunately, ‘La Quintrala’ is a musical composition and non-utilitarian, and not an analysis tool. Handling the expressive data became an issue of finding the best possible way to have the data influence and enhance the soundscape, based on my subjective, compositional judgment.

Exactly what kind of expressive data I needed to access for would vary considerably throughout the opera. The vague definition of 'expressive data' centers on flexible timing and flexible use of dynamics and articulation, and we often refer to this important part of music performance as 'phrasing'. A vast variety of imprecise musical terms is accounted for in score notation concerning loudness, pitch connection, articulation, timbral shading, and time contraction and expansion etc., such as mezzo-forte, legato, tenuto, lontano, rallentando, and so forth. Since these notations are not as easily measurable as pitch and time, they are considered to be in the domain of performers, and a performance is judged by how well the performer expressively alters the given pitches and rhythms within the nature of the composition. Paradoxically, this expressive layer carries much of the appreciable content of the composition, because the composition defines and delimits its own 'expressive space', as an implicit consequence of the composer's decisions of the more precise notations of pitch, time and rhythm. The 'expressive layer' is therefore a hidden but integral part of any musical composition, and even though it is defined at the moment of composing it only comes only into existence at the moment of performance. We

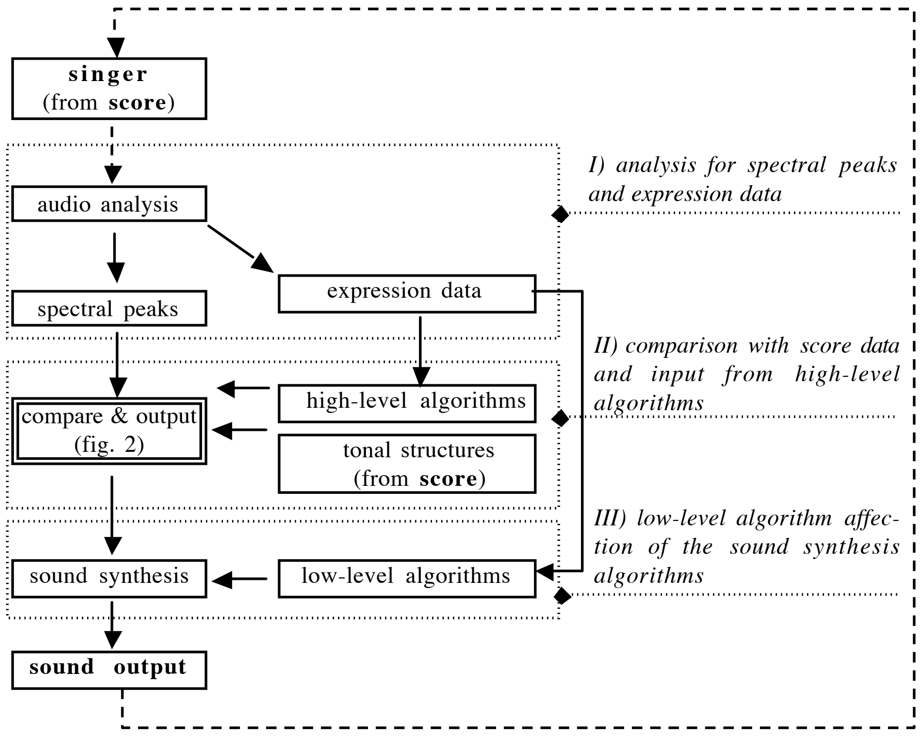

singer (from score)
audio analysis
spectral peaks
expression data
compare & output (fig. 2)
high-level algorithms
tonal structures (from score)
sound synthesis
low-level algorithms
sound output
I) analysis for spectral peaks and expression data
II) comparison with score data and input from high-level algorithms
III) low-level algorithm affection of the sound synthesis algorithms

Fig. 1. Overall data fow (dashed arrows represent sound)

rely on performers to add this layer to their interpretation of a score, and it would be of major importance for me to extract and apply in some form this expressive data to the accompanying soundscape throughout the opera.

5 The System

The aim for the interactive relationship between the singers and the composition in 'La Quintrala' had become twofold: comparing pitches for chordal verification and support, and projecting the singer's interpretation of the expressive parameters onto the accompanying soundscape.

The system is divided into three parts, I) analysis for spectral peaks and expression data, II) comparison with score data and input from high-level algorithms, and III) low-level algorithm affection of the sound synthesis algorithms (fig. 1).

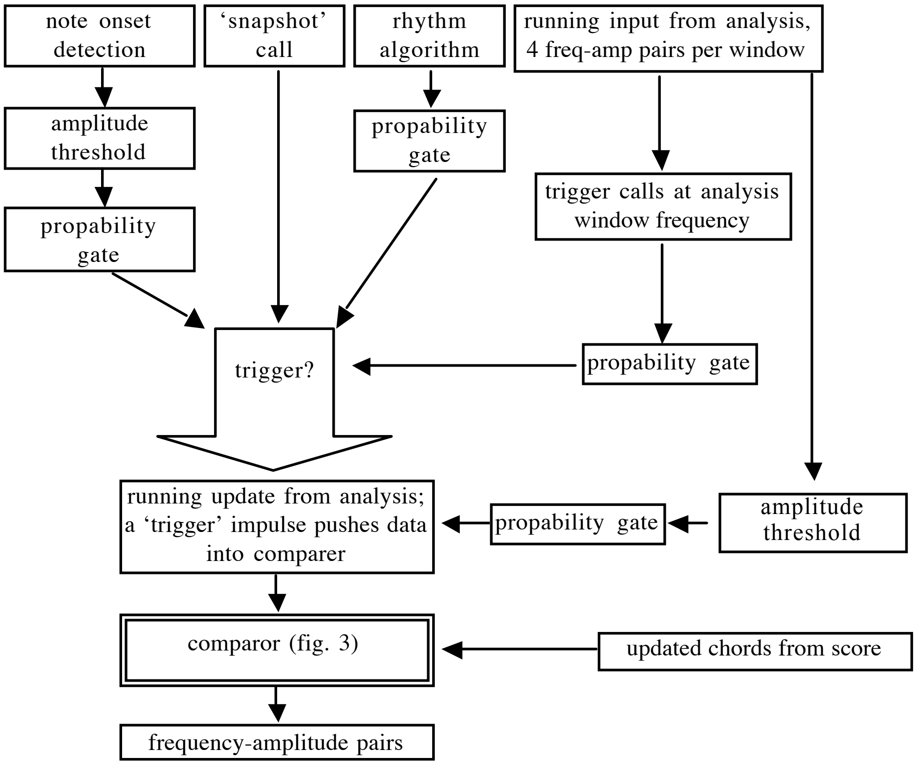

note onset detection
'snapshot' call
rhythm algorithm
running input from analysis, 4 freq-amp pairs per window
amplitude threshold
propability gate
trigger calls at analysis window frequency
propability gate
trigger?
propability gate
running update from analysis; a 'trigger' impulse pushes data into comparer
propability gate
amplitude threshold
comparor (fig. 3)
updated chords from score
frequency-amplitude pairs

Fig. 2. Input-output flow for comparison of spectral peaks

5.1 Analysis

The analysis phase outputs the four most prominent spectral peaks as frequency-amplitude pairs, an impulse upon note-on detection, a continuous amplitude envelope and a fundamental pitch estimation. The original algorithm used is described in [30]. The spectral peaks are gated with an amplitude threshold and a probability gate, and the running update can – but may not necessarily, depending on the settings of these gates – store new values at analysis window frequency, for possible passing on to the comparer.

5.2 Data Triggering

Input to the comparer is triggered in parallel by four methods, individually controlled in terms of density, and combined into one output (fig. 2):

1. analysis window frequency,
2. a rhythmizised trigger,
3. by note onset detection, and
4. single comparison trigger (a 'snapshot'), called from within the application.

Triggering at analysis window frequency is basically a simple, additive resynthesis approach with four waveforms. The rythmizised trigger is produced by an algorithm generating onset pulses at timed intervals, with control of regularity of the pulse and probability of the trigger impulse taking place. Note onset triggering is determined by changes in amplitude or fundamental and passed through a threshold and probability gate, while single comparison triggers are called from within the application when required.

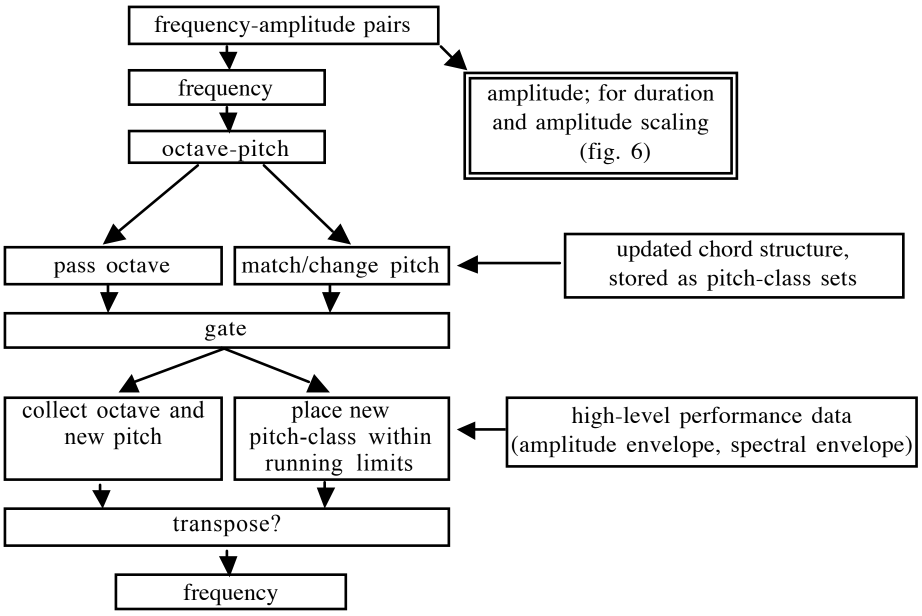

frequency-amplitude pairs
frequency
octave-pitch
amplitude; for duration and amplitude scaling (fig. 6)
pass octave
match/change pitch
updated chord structure, stored as pitch-class sets
gate
collect octave and new pitch
place new pitch-class within running limits
high-level performance data (amplitude envelope, spectral envelope)
transpose?
frequency

Fig. 3. The comparor

When a trigger is effectuated, it is pushes the most recent frequency-amplitude pairs received from the running analysis into the comparer for parsing, which in turn will return new frequency-amplitude pairs.

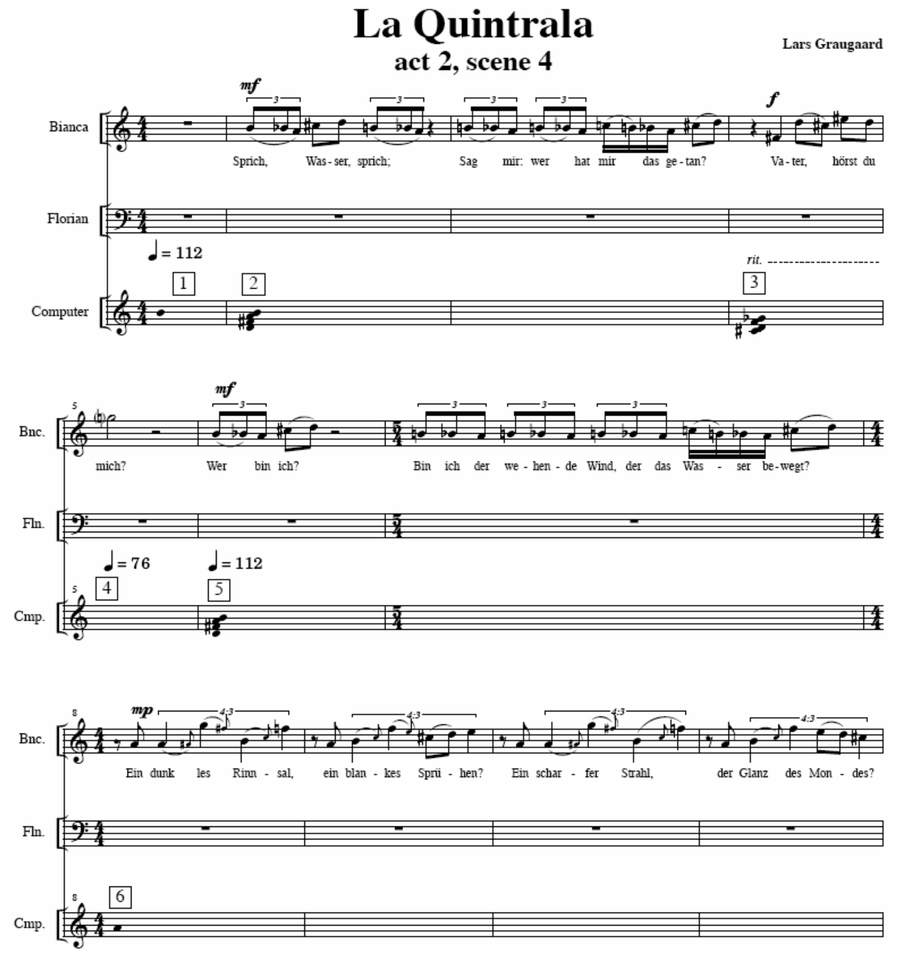

La Quintrala
act 2, scene 4
Lars Graugaard
Bianca
Florian
Computer
Bnc.
Fln.
Cmp.
♩ = 112
♩ = 76
rit.
Sprich, Was-ser, sprich; Sag mir: wer hat mir das ge-tan? Va-ter, hörst du
mich? Wer bin ich? Bin ich der we-hen-de Wind, der das Was-ser be-wegt?
Ein dunk les Rinn-sal, ein blan-kes Sprü-hen? Ein schar-fer Strahl, der Glanz des Mon-des?
1
2
3
4
5
6

Fig. 4. Score example from La Quintrala, third staff is supporting pitch-classes

5.3 Data Comparison

The second part parses the triggered frequency-amplitude pairs based on the stored pitch-classes and a set of parameters. The frequencies from the analyzer are converted to midi pitches and octave and pitch class are separated (fig. 3). The octave is passed, and the pitch is matched to a reference pitch-class set. This pitch-class set is stored in memory, and passed to the comparer as the events in the score are advanced by hand in performance.

The stored pitch-classes define which pitch-classes are the basis of the melodic material at those particular bars of the accompanying score (fig. 4). The pitch-class sets have been defined beforehand as a chordal structure that supports a given subsection of the melody. I derived these chordal structures from the melodic material they supported, which in turn was developed from the requirements of the libretto in terms of dramatic content and musical needs. The structures were made with an informal technique of floating pitch connection, with no functional harmony. The chords can be seen as chroma, i.e. pitch-classes independent of octave placement and inversion, and each chord usually delimits a time segment corresponding to a melodic segment that can be sustained by a set of up to five pitches.

The comparison produces a pitch-class output describing the best musical match to accompany the singing. The best match is modified by an index so as to make it possible to vary the consonance/dissonance relation between singer and soundscape/accompaniment, since the best musical match for compositional reasons wouldn't necessarily be the most consonant match. The resulting pitch-class is then either re-collected with the octave information, or the octave information is discarded and the pitch-class is placed within a running low-high pitch limit. The harmonic structures become 'associative' as pitch centers attract the melody and thereby is perceived to shape and guide the melodic contours. Equation (1) handles this, as well as micro-tonal deviations according to a dissonance factor.

$$\begin{gathered} y = \mathrm{int}(x) + d{*}x \bmod 1 \text{ for } 0. \leq x \bmod 1 < 0.5 \\ y = \mathrm{int}(x) + 1 - d{*}\mathrm{abs}(x \bmod 1 - 1) \text{ for } 0.5 \leq x \bmod 1 < 1\,. \end{gathered} \tag{1}$$

where d is dissonance factor.

The comparisor handles pitch-classes as a container of microtonal pitch information 0.5 semitones above and below its center. Applying gradual controls to these containers makes it possible to 'pull in' the generated frequencies towards the stored chord, or to 'release' them. This works well in conjunction with running pitch output, even though it defeats the tonal precision of the accompaniment.

The running low-high pitch limit is correlated to the running amplitude envelope with parameters for center, follow, and range (equation 2). The center value follows the amplitude envelope either directly or inversely according to the follow value, which specifies the center offset at maximum amplitude.

$$\begin{gathered} c + (a{*}f) + 0.5{*}r = \text{high limit} \\ c + (a{*}f) - 0.5{*}r = \text{low limit}\,. \end{gathered} \tag{2}$$

where c is pitch center, r is pitch range, a is amplitude envelope, and f is the follow factor expressed as the ratio pitch-steps/max-amplitude.

The singer can hereby affect the pitch range of the output of the comparer in performance, and the code for equations (1) and (2) can be seen in (fig. 5).

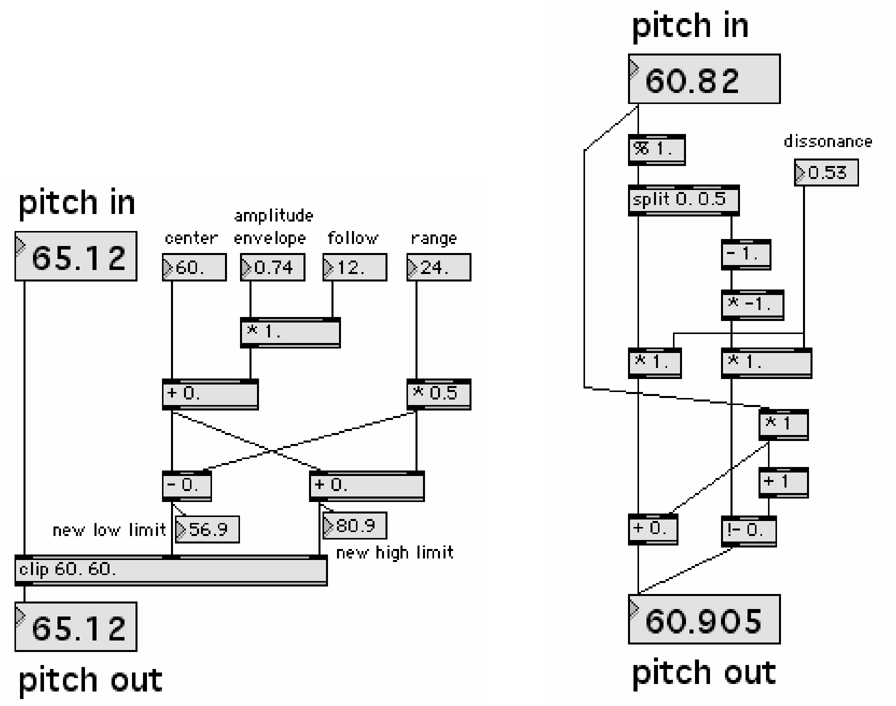


Fig. 5. Code examples of equations (1) and (2)

5.4 Amplitude as Expressive Parameter

The amplitude obtained during analysis is also applied to note duration and base amplitude of the corresponding frequency-amplitude pair from the comparer (fig. 6). It made sense to have the ability to apply amplitude envelopes to the comparison results, because this would make it possible to affect sound synthesis taking place elsewhere in the application, provided it recognized such data. It would serve the need for subjective, dynamic mapping of expressive data according to the evolving musical needs as I perceived them. Such expressive data doesn't interfere with the pitch estimation algorithm but affects the duration and amplitude of each comparison result on an ad-hoc basis (as well as the pitch range, as described by equation 2). High-level parameters were devised for generalized control, so that duration information was submitted to a scaling value where positive scaling would yield longer durations for higher amplitudes, zero would be unity duration, while negative scaling value would yield shorter durations.

5.5 Direct Audio Output

Finally, a direct output was added whereby each comparison result would be played back as an audio waveform, using the above mentioned amplitude envelope.

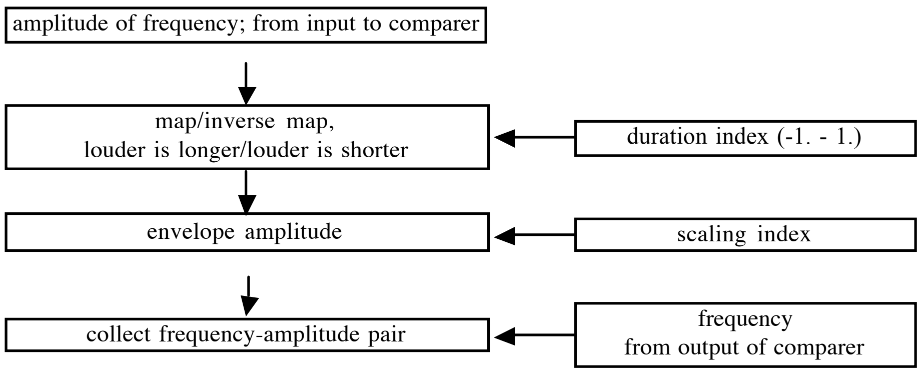

amplitude of frequency; from input to comparer
map/inverse map, louder is longer/louder is shorter
duration index (-1. - 1.)
envelope amplitude
scaling index
collect frequency-amplitude pair
frequency from output of comparer

Fig. 6. Duration scaling by amplitude

Parameters are exposed for switching waveform (not necessarily being a sinusoid), setting the amplitude envelope, and for spatial placement in a four channel XY-field.

5.6 The System in Performance

The system shapes the accompaniment by some combination of stored information pertaining the requirements of the composition together with the needs of the singers for vocal cueing, and the dramatic content of the moment as it develops on stage in interaction between the characters. As a result of this combined approach, the accompanying electronic soundscape in ‘La Quintrala’ could reconcile formal, compositional needs for structure and tonal support with sonic manifestation of immediate emotion expressed during performance by the singers. Furthermore, the possibility to make gradual transitions between these two requirements provided flexibility in handling transitions between different musical contexts. The combination of sound algorithm and analysis data affecting the spectral content of the synthesis proved an effective way to enhance the presence of the performer in the electronic accompaniment and project the singer’s musicality into the accompaniment. In fact, the accompaniment in ‘La Quintrala’ was very often described to me by members of the audience as an immediate, aural manifestation of the psychological disposition of the singers, and of the emotional charge of a scene.

The singers’ expressivity and musicality projected into the electronics of ‘La Quintrala’ is one of interplay rather than actual control, since the algorithms driving the sound synthesis as well as my decisions on handling the correlation exceptions of analysis output versus stored chordal structure are partially hidden to them. This should not cause concern, because interaction implies some degree of lack of control on part of the musical performer [25]. The purpose of giving presence to the singers did not coincide with giving (full) control over the electronic score, since the drama being developed isn’t determined by such local action-reaction mechanisms, but by the larger-scale dealings and consequences. The singers’ psychological dispositions are manifested and evolves in the electronics, and the resulting expansion – or intensification – of the dramatic content multiplies the emotional substance in a way very appropriate to opera.

6 Research Background

The system relates primarily to score following, signal decomposition, performer-computer interaction, interactive music accompaniment systems, and presence enhancement and mood creation.

6.1 Signal Decomposition

The score correlation method presented doesn't include original work on signal decomposition, but it was important to choose the proper method for audio analysis. The algorithm used is described in [30], and subsequently AltiVec optimized and further developed by Jehan et al. [20], even though the authors don't specify how the algorithm was enhanced. The analysis algorithm outputs doesn't distinguish between sinusoidal and noise component, but it performs reasonably well for my purpose, since the signal analyzed in 'La Quintrala' is known to be a – mostly – periodic signal.

Signals with a large noise component or in circumstances of low signal-to-noise ratio are not well handled by pure spectral peak estimation. A procedure is proposed in [16] for preanalyzing a signal for spectral peaks corresponding to true sinusoidal components, whereby the influence of noise on the analysis result is considerably reduced. The technique seems presently too slow for real time application. An approach for classifying the spectral peaks into noise and sinusoidal peaks is presented in [31], where the authors suggest to limit the number of candidate peaks in order to reduce the computational cost. This approach is relevant to my purpose, as I only need a limited number of peaks.

Conventional content-based audio representations use statistical characteristics, but this has limitations in terms of content representation as described in [26]. A content-based retrieval system has been described in [1] and the authors reported experiments which show that the singular value of the 'first principle component' usually is greatly higher than others for the purpose of general feature extraction. This is comparable to the pitch estimation in terms of precision of perception, and accommodates for fluctuations resulting from the expressive musicality of the performance. Such analysis methods would enhance the decomposition for extracting high-level expressivity data.

The need for real time analysis puts restrictions on the algorithms. Not all approaches mentioned are therefore feasible at this moment, even if they suggest results that could be very useful for my purpose. But given the development in computational power and in the algorithms themselves, it must be expected that real time analysis methods will develop further features such as those mentioned.

6.2 Score Following

Score correlating in 'La Quintrala' has the purpose of cueing and tonally supporting a singer, by relating the spectral content to a stored chord, and to sonically enhance the presence of the singer in the accompaniment. Score correlating is related to score following, and it was indeed my intention to include score following functionality in 'La Quintrala'. Score following gives score position and future tempo estimate, for advancing the computer's reading of the electronic score in synchronization with the performance of the music. For this purpose score following attempts to transform the performance into a real-time sequence of note onsets and their corresponding discrete pitch.

Score following was first presented by Barry Vercoe and Roger Dannenberg in 1984 [34, 9, 10]. Later systems pioneered at IRCAM [28] tried to minimize system latency rather than predicting tempo, since steadfast tempo often would be absent from performance, or possibly only vaguely implied. One of the first examples of these efforts was Philippe Manoury's composition 'Jupiter' for flute and interactive computer. It consists of a great many events to be automatically triggered through the proper advancing by the score following algorithm. This works fairly well, but not sufficiently reliable to avoid human supervision during performance. But while the flute is a relatively simple signal to track, it was of concern to me that the singing voice is a particularly difficult subject for such detection [28]. The voice is an extremely flexible 'instrument' characterized by timbral and dynamic richness – the voice has a dynamic range of up to 55dB – and capable of a great many expressive performance parameters such as vibrato and glissandi, let alone the particular influence of the sung text. All these performance 'artifacts' make fast and accurate onset and pitch detection for score following purposes unreliable, if at all possible. For this reason any score following functionality was eventually left out of 'La Quintrala', since the purpose was to rely solely on the follower to advance the events.

A method for tracking the score position of a singer based on stochastic procedures was later presented in [14]. In order to increase reliability, the authors suggested to extend the score following model to include features other than fundamental pitch. Tracking through stochastic modeling a variety of performance data using hidden markov models was presented in [7] and [27]. In the latter system, the authors treated the problem as a subclass of sequence alignment, and expanded on techniques first developed in speech recognition and in molecular genetics. A technique for use with nonlinear, open-form score notation has been described in [23]. The follower was mainly used for audio routing and mixing purposes, based on the 'crossroad' concept known from some open-form compositions. The linearity of standard score notation was avoided by using a noise-reduced confidence-index to locate the tracker in the sections. The index would then be used for making global changes at signal routing and mixing level, but also of signal processing algorithms.

Score following in its extreme is an automated accompaniment system, and attempts have been made at refining and commercializing such systems. [18], [21], and [19] describe systems that accompany amateur vocalists performing pop music. The first two rely on pitch detection for tracking the performer, while the last applies speech processing techniques for vowel recognition. The systems attempt to identify both the score position and the tempo of the performer, and to adjust the computer accompaniment in response. CODA Music Group introduced with SmartMusic™ a commercial accompaniment system for amateur musicians.

The latest score following algorithms have reached a fair degree of accuracy, but the somewhat more simple score following algorithms I considered for 'La Quintrala' are prone to errors. On the one hand the performer may make errors in performance, while real time audio analysis algorithms on the other hand aren't fail-safe concerning pitch estimation. And if the musician falters then it is of little comfort if the following algorithm works correctly. Consequently, it becomes mandatory to compose for easier performance and score following, such as avoiding writing which cannot be accurately followed because it doesn't translate easily into simple notation, or avoid passages which are of great difficulty in interpretation or execution. The compositional consequences of these stability issues made me refrain from including score following in 'La Quintrala': it would limit the score notation unacceptably (and hence the musical possibilities), and it seemed moot when the performance in any case would have to be supervised, due to the risk of failings of the tracking.

6.3 Mood Creation and Presence Enhancement

Most score following systems are concerned with extracting pitch and duration of a musical performance. Yet expressivity is one of the salient features in music appreciation. It has been my intention to map expressiveness onto the electronic soundscape through mapping salient performance aspects onto high-level output parameters. Observations are made in [8] concerning how parametrization is capable of trivializing or enhancing interactivity in a human-machine relationship, and mood creation through adding expressiveness to an automatic musical performance is described in [4]. In [3] a performance was analyzed and the data applied to a computer-generated performance, resulting in robust detection of the emotional intention by expert listeners. It could therefore be considered to map such data onto synthesis algorithms, and signal routing and mixing.

Real time emotion tracking is described in [13], where a limited set of cues can accurately predict a set of emotional expressions, without using any score information. The system compares the extracted cues to a previously stored reference input, and the strong intercorrelation of the cues in a given emotion makes for quite accurate prediction. The authors use the output for a graphical representation of the intended emotions, but this data could just as well be used for high-level control of synthesis algorithms, and signal routing and mixing. Another system, designed to work with MIDI instruments, is described in [6]. The basis is a 'Perceptual Parametric Space' described in [5] which relate sets of coefficients to acoustic quantities. The system would require accurate fundamental pitch estimation, note duration, etc. in order to work with an audio signal.

Emotion as acquired sensibility towards art and music is presented with reference to its Japanese word 'kansei' in [17]. The author describes this as the third target of information processing referring to feelings, intuition, and sympathy, while the second target is semantic symbol processing, and the first target being the physical signal. Contributions in the area of 'kansei' encompassing not only music are described in ao. [2]. Emotion can be broken down to a fairly limited set of parameters, and following these parameters have proven to give fairly precise results for predicting the emotion intended. I have tentatively interpreted selected performance parameters and projected them onto synthesis algorithms, signal routing and mixing, and other aspect of an expressive space. This seems as a fertile area which requires a fine balance between the trivializing and enhancing, the obvious and obscure, and the traceable and multidimensional [24].

7 Conclusions, Further Development, and Other Applications

The described performer-accompaniment unification approach proved very effective in musical drama. Projecting emotions into an accompanying soundscape as they are exposed by the singers on stage enhances the audience's appreciation of their presence and mood which in turn enhances the dramatic content and development, strongly engaging the audience's attention.

A content-based system for analyzing the audio as suggested will be considered.

Interactive instrumental music without dramatic action may contain a high emotional, yet abstract impact, even though the voice is the one instrument which offers the widest range of possible variation in timbre. Interactive instrumental music without dramatic action still has a high emotional import which isn't referred to any object or objective. This presumes that we accept that music really is a language of emotion, primarily expressing the composer's knowledge of human feeling, as expressed in [22]. The performer-accompaniment unification approach seems therefore readily adaptable to interactive instrumental music as well, where attaching performance expressivity to the electronic soundscape in parallel with more 'autonomous' evolution of the soundscape could be used to advantage. I therefore intend to apply the technique of expressive projection in interactive, instrumental music, and expect to find further development possibilities in this area, and most likely quite different from those explored in 'La Quintrala'.

References

1. Cai, R., Lu, L., Zhang, H-J., and Cai, L-H.: Improve Audio Representation by Using Feature Structure Patterns. ICASSP04 (2004).
2. Camurri, A., Trocca, R., Volpe, G.: Interactive systems design: A KANSEI-based approach. In: Proceedings of NIME2002. Dublin, Ireland (2002).
3. Camurri, A., Dillon, R., Saron, A.: An Experiment on Analysis and Synthesis of Musical Expressivity. In: Proceedings of the XIII Colloquium on Music Informatics. L'Aquila, Italy (2000).
4. Canazza, S., Rodá, A.: Adding Expressiveness in Musical Performance in Real Time. In: Proceedings of the AISB 1999, Symposium on Musical Creativity, pp. 134-139. Edingburgh, Scotland (1999).
5. Canazza, S., De Poli, G., Rodá, A., Vidolin, A., Zanon, P.: Kinematics-energy space for expressive interaction in music performance. In: Proceedings of MOSART, Workshop on current research directions in Computer Music, pp. 35-40. Barcelona, Spain (2001).
6. Canazza, S., De Poli, G., Rodá, A., Soleni, G., Zanon, P.: Real Time Analysis of Expressive Contents in Piano Performances. In: Proceedings of the 2002 International Computer Music Conference, pp. 414-418. Gothenburg, Sweden (2002).
7. Cano, P., Loscos, A., Bonada, J.: Score-performance matching using HMMs. In: Proceedings of the 1999 International Computer Music Conference. Beijing, China (1999).
8. Choi, I.: Interactivity vs. control: human-machine performance basis of emotion. In: Proceedings of 1997 'KANSEI - The Technology of Emotion Workshop', pp. 24–35 (1997).

9. Dannenberg, R. and Mukaino, H.: New Techniques for Enhanced Quality of Computer Accompaniment. In: Proceedings of the 1988 International Computer Music Conference, pp. 243-249. Cologne, Germany (1988).
10. Dannenberg, R.: An On-line Algorithm for Real-Time Accompaniment. In: Proceedings of the 1984 International Computer Music Conference pp. 193-198. Paris, France (1984).
11. Forte, A.: The Structure of Atonal Music. Yale University Press. New Haven and London (1973).
12. Freescale Semiconductor: Altivec technology; www.simdtech.org/altivec
13. Friberg, A., Schoonderwaldt, E., Juslin, P. N., Bresin, R.: Automatic Real-Time Extraction of Musical Expression. In: Proceedings of the 2002 International Computer Music Conference, pp. 365-367. Gothenburg, Sweden (2002).
14. Funkhauser, T., Jot, J-M, Tsingos, N.: Sounds Good to Me. SIGGRAPH 2002 Course Notes (2002).
15. Grubb L., Dannenberg, R.: A stochastic method of tracking a vocal performer. In: Proceedings of the 1997 International Computer Music Conference, pp. 301-308. Thessaloniki, Greece (1997).
16. Graugaard, L.: La Quintrala. Opera for five singers and interactive, computer generated accompaniment. Premiered September 2nd 2004 at Den Anden Opera, Copenhagen, Denmark (2003-04).
17. Hanna, P., Desainte-Catherine, M.: Detection of sinusoidal components in sounds using statistical analysis of intensity fluctuations. In: Proceedings of the 2002 International Computer Music Conference, pp. 100-103. Gothenburg, Sweden (2002).
18. Hashimoto, S.: KANSEI as the Third Target of Information Processing and Related Topics in Japan. In: A.Camurri (Ed.) Proceedings of the International Workshop on KANSEI: The Technology of Emotion, pp.101-104 (1997).
19. Inoue, W., Hashimoto, S., Ohteru, S.: A computer music system for human singing. In: Proceedings of the 1993 International Computer Music Conference, pp. 150-153; Tokyo, Japan (1993).
20. Inoue, W., Hashimoto, S., Ohteru, S.: Adaptive karaoke system—human singing accompaniment based on speech recognition. In: Proceedings of the 1994 International Computer Music Conference, pp. 70-77. Århus, Denmark (1994).
21. Jehan, T., Schoner, B.: An Audio-Driven Perceptually Meaningful Timbre Synthesizer. In: Proceedings of the 2001 International Computer Music Conference. Havana, Cuba, (2001)
22. Katayose, H., Kanamori, T., Kamei, K., Nagashima, Y., Sato, K., Inokuchi, S., Simura, S.: Virtual performer. In: Proceedings of the 1993 International Computer Music Conference, pp. 138-145. Tokyo, Japan (1993).
23. Langer, S. K.: Philosophy in a New Key. Harvard University Press. New American Library, New York, New York
24. Lawter, J., Moon, B.: Score following in open form compositions. In: Proceedings of the 1998 International Computer Music Conference,. San Francisco, USA (1998).
25. Lippe, C.: A Look at Performer/Machine Interaction Using Real-Time Systems. In: Proceedings of the 1996 International Computer Music Conference, pp. 116-117. Hong Kong (1996).
26. Lippe, C.: Real-Time Interaction Among Composers, Performers, and Computer Systems. In: Information Processing Society of Japan SIG Notes, Volume 2002, Number 123, pp. 1-6. (2002)
27. Martin, K. D., Scheirer, E. D., and Vercoe, B. L.: Music Content Analysis through Models of Audition. ACMMM98 (1998).

28. Orio, N., Lemouton, S., Schwarz, D.: Score following: state of the art and new developments. In: Proceedings of the 2003 Conference on New Interfaces for Musical Expression. Montreal, Canada (2003).
29. Puckette, M., Lippe, C.: Score following in practise. In: Proceedings of the 1992 International Computer Music Conference, pp. 182-185. (1992)
30. Puckette, M.,: Score following using the sung voice. In: Proceedings of the 1995 International Computer Music Conference. Banff, Canada (1995).
31. Puckette, M., Apel, T., Zicarelli, D.: Real-time audio analysis tools for Pd and MSP. In: Proceedings of the 1998 International Computer Music Conference, pp. 109-112. San Francisco, USA (1998).
32. Roebel, A., Zivanoviz, M., Rodet, X.: Signal decomposition by means of classification of spectral objects. In: Proceedings of the 2004 International Computer Music Conference, pp. 446-449. Florida, USA (2004).
33. Rowe, R.: Machine Musicianship, pp. 191-193. MIT Press, ISBN 0-262-68149-8 (2001).
34. Vercoe, B.: The Synthetic Performer in the Context of Live Performance. In: Proceedings of the 1988 International Computer Music Conference, pp. 199-200. Paris, France (1998).

Recognizing Chords with EDS: Part One

Giordano Cabral[1], François Pachet[2], and Jean-Pierre Briot[1]

[1] Laboratoire d'Informatique de Paris 6,
8 Rue du Capitaine Scott, 75015 Paris, France
{Giordano.CABRAL, Jean-Pierre.BRIOT}@lip6.fr
[2] Sony Computer Science Lab,
6 Rue Amyot, 75005 Paris, France
pachet@csl.sony.fr

Abstract. This paper presents a comparison between traditional and automatic approaches for the extraction of an audio descriptor to recognize chord into classes. The traditional approach requires signal processing (SP) skills, constraining it to be used only by expert users. The Extractor Discovery System (EDS) [1] is a recent approach, which can also be useful for non expert users, since it intends to discover such descriptors automatically. This work compares the results from a classic approach for chord recognition, namely the use of KNN-learners over Pitch Class Profiles (PCP), with the results from EDS when operated by a non SP expert.

1 Introduction

Audio descriptors express by mathematical formula a particular property of the sound, such as the tonality of a musical piece, the amount of energy in a given moment, or whether a song is instrumental or sung. Although the creation of each descriptor requires a different study, the design of a descriptor extractor normally follows the process of combining the relevant characteristics of acoustic signals (features) using machine learning algorithms. These features are often low-level descriptors (LLD), and the task usually requires important signal processing knowledge.

Since 2003, a heuristic-based approach became available through the Computer Science Lab of Sony in Paris, which developed the Extractor Discovery System (EDS). The system is based on genetic programming, and machine learning algorithms employed to automatically generate a descriptor from a database of sound files examples and their respective perceptive values. EDS can be used either by non experts or expert users. Non experts can use it as a tool to extract descriptors, even with minimal or no knowledge at all in signal processing. For example, movie makers have created classifiers of sound samples to be used in their films (explosions, car breaks, etc.). Experts can use the system to improve their results, starting from their solution and then controlling and guiding EDS. For instance, the perceived intensity of music titles can be more precisely revealed, taking as a starting point the mpeg7 audio features [2].

We are currently designing a guitar accompanier for "bossa nova" style. During the application development process, we ran into the problem of recognizing a chord, which turned out to be a good opportunity of comparing classical and EDS approaches. On the one hand, chord recognition is a well studied domain, with solid results that can be considered as reference. On the other hand, current techniques use background knowledge that EDS (initially) does not have (pitches, harmony). Good EDS results would indicate the capacity of the system to deal with real world musical description cases.

We intend to compare the results from a standard technique of chord recognition (KNN learner over Pitch Class Profiles) and those from EDS, when operated by an inexperienced user (so called Naïve EDS) and by an expert user (so called Expert EDS). This paper presents the first part of this comparison, considering only the results obtained by the Naïve EDS. In the next section, we introduce the chord recognition problem. In section 3 we explain the most widely used technique. In section 4 we examine EDS, how it works and how to use it. Section 5 details the experiment. Section 6 shows and discuss the results. Finally, we draw some conclusions and point future works.

2 Chord Recognition

The ability of recognizing chords is important for many applications, such as interactive musical systems, content-based musical information retrieval (finding particular examples, or themes, in large audio databases), and educational software. Chord recognition means the transcription of a sound into a chord, which can be classified according to different levels of precision, from a simple distinction between maj and min chords to a complex set of chord types (maj, min, 7th, dim, aug, etc).

Many works can be mentioned here as the state of the art in chord recognition. [4] and [5] automatically transcribes chords from a CD recorded song. [3] deals with a similar problem: estimating the tonality of a piece (which is analogous to the maj/min). In most cases the same core technique is used (even if some variations may appear during the implementation phase): the computation of a Pitch Class Profile, or *chromagram*, and a subsequent machine learning algorithm to find patterns for each chord class. This technique has been applied to our problem, as we explain in next section.

3 Traditional Technique: Pitch Class Profiles

Most part of the works involving harmonic content (chord recognition, chord segmentation, tonality estimation) uses a feature called Pitch Class Profile (PCP) [6]. PCPs are vectors of low-level instantaneous features, representing the intensity of each pitch of the tonal scale mapped to a single octave. These vectors are calculated as follows: 1) a music recording is converted to a Fourier Transform representation (Fig1a to Fig1b). 2) the intensity of a pitch is calculated (Fig1b to Fig1d) by the magnitude of the spectral peaks, or by summing the magnitudes of all frequency bins that are located within the respective frequency band (Fig1c). 3) The equivalent pitches from different octaves are summed, producing a vector of 12 values (eventually 24 to deal with differences in tuning and/or to gain in performance), consequentially unifying various dispositions of a single chord class (Fig1e and Fig1f). For example, one can expect that the intensities of the frequencies corresponding to the notes C, E and G in the spectrum of a Cmaj would be greater than the others, independently on the particular voicing of the chord.

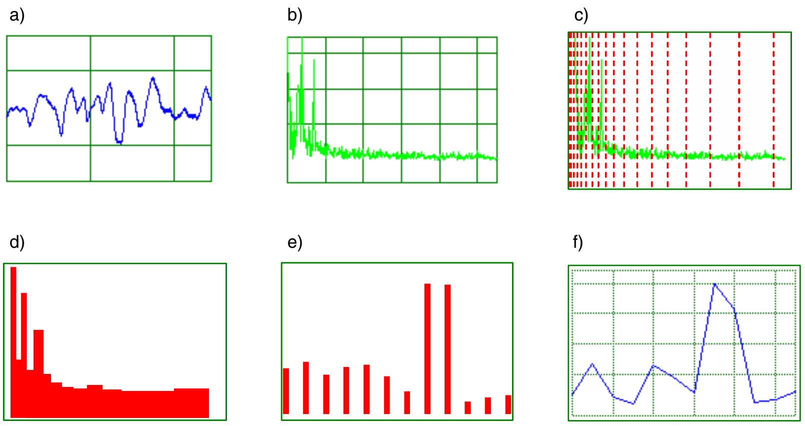


Fig. 1. Steps to compute a PCP. The signal is converted to Fast-Fourier representation; the FFT is divided into regions; the energy of each region is computed; the final vector is normalized.

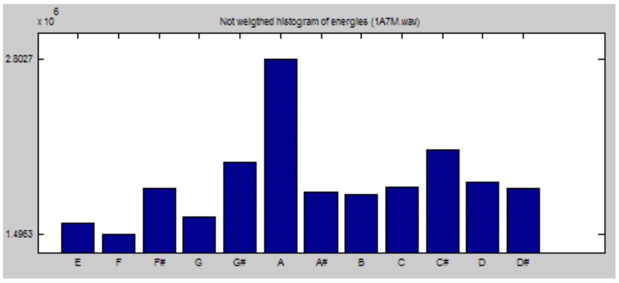


Fig. 2. Example of the PCP for a Amaj7. Each column represents the intensity of a note, independently on the octave.

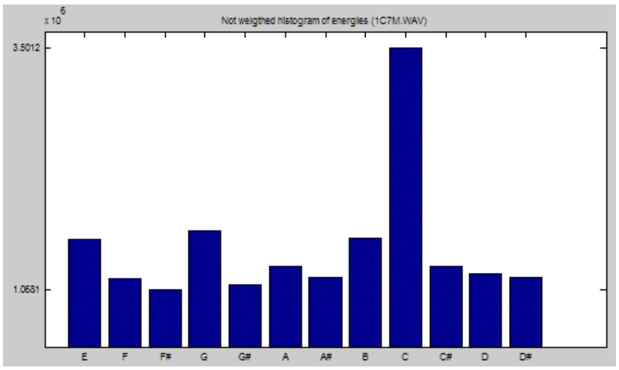


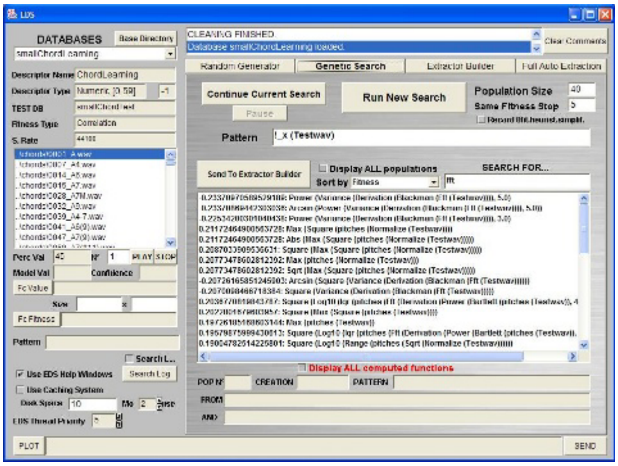
Fig. 3. Example of the PCP for a Cmaj7

The idea of using PCPs to chord recognition is that the PCPs of a chord follow a pattern, and that patterns can be learned from examples. Thus, machine learning (ML) techniques [9] can be used to generalize a classification model from a given database of labeled examples, in order to automatically classify new ones. So, for the PCP of a chord, the system will respond the most probable (or closest) chord class, given the examples previously learned. The original PCP implementation from Fujishima used a KNN learner [6], and more recent works [3] successfully used other machine learning algorithms.

4 EDS

EDS (Extractor Discovery System), developed at Sony CSL, is a heuristic-based generic approach for automatically extracting high-level music descriptors from acoustic signals. EDS is based on Genetic Programming [11], used to build extraction functions as compositions of basic mathematical and signal processing operators, such as *Log*, *Variance*, *FFT*, *HanningWindow*, etc. A specific composition of such operators is called feature (e.g. *Log (Variance (Min (FFT (Hanning (Signal)))))*), and a combination of features form a descriptor.

Given a database of audio signals with their associated perceptive values, EDS is capable to generalize a descriptor. Such descriptor is built by running a genetic search to find relevant signal processing features to match the description problem, and then machine learning algorithms to combine those features into a general descriptor model.



```
smallChordLearning
Descriptor Name  ChordLearning
Descriptor Type  Numeric [0 59]  -1
TEST DB          smallChordTest
Fitness Type     Correlation
S. Rate          44100
.\chords\C001_A.wav
.\chords\C007_A4.wav
.\chords\C014_A6.wav
.\chords\C016_A7.wav
.\chords\C028_A7M.wav
.\chords\C032_A9.wav
.\chords\C039_A4-7.wav
.\chords\C041_A6(9).wav
.\chords\C047_A7(9).wav
Perc Val 40   N° 1
Model Val     Confidence
Size   x
Pattern
☐ Search L...
☑ Use EDS Help Windows
☐ Use Caching System
Disk Space 10   Mo  2   Use
EDS Thread Priority 5

CLEANING FINISHED.
Database smallChordLearning loaded.
Population Size 40
Same Fitness Stop 5
☐ Record fit.heurist.simplif.
Pattern  !_x (Testwav)
☐ Display ALL populations        SEARCH FOR...
Sort by Fitness                  fft
-0.23370970589529108: Power (Variance (Derivation (Blackman (Fft (Testwav)))), 5.0)
0.23370889442303038: Arcsin (Power (Variance (Derivation (Blackman (Fft (Testwav)))), 5.0))
-0.22534200301040438: Power (Variance (Derivation (Blackman (Fft (Testwav)))), 3.0)
0.21172464900563728: Max (Square (pitches (Normalize (Testwav))))
0.21172464900563728: Abs (Max (Square (pitches (Normalize (Testwav)))))
0.20870339095536631: Square (Max (Square (pitches (Normalize (Testwav)))))
0.20773478602812392: Max (pitches (Normalize (Testwav)))
0.20773478602812392: Sqrt (Max (Square (pitches (Normalize (Testwav)))))
-0.20726165851245803: Arcsin (Square (Variance (Derivation (Blackman (Fft (Testwav))))))
-0.2070098466718384: Square (Variance (Derivation (Blackman (Fft (Testwav)))))
0.2038770819843787: Square (Log10 (Iqr (pitches (Fft (Derivation (Power (Bartlett (pitches (Testwav)), 4
0.20220016786803857: Square (Max (Square (pitches (Testwav))))
0.197261805168603144: Max (pitches (Testwav))
0.19679875989430813: Square (Log10 (Iqr (pitches (Fft (Derivation (Power (Bartlett (pitches (Testwav)),
0.19004782614225801: Square (Log10 (Range (pitches (Sqrt (Normalize (Testwav))))))
☐ Display ALL computed functions
POP N°   CREATION   PATTERN
FROM
AND
```

Fig. 4. EDS main interface

The genetic search performed by the system is intended to generate functions that may eventually be relevant to the problem. The best functions in a population are selected and iteratively transformed (by means of reproduction, i.e., constant variations, mutations, and/or cross-over), always respecting the pattern chosen by the user. The default pattern is *!_x(Signal)*, which means a function presenting any number of operations but a single value as result. The populations of functions keep reproducing until no improvement is achieved. At this point, the best functions are selected to be combined. This selection can be made both manually or automatically. For example, given a database of audio files labeled as 'voice'/'instrumental', kept the default pattern, these are some possible functions that might be selected by the system:

```
Log10 (Range (Derivation (Sqrt (Blackman (MelBands (Signal, 24.0))))))

Square  (Log10  (Mean  (Min  (Fft (Split (Signal, 4009))))))
```

Fig. 5. Some possible EDS features for characterizing a sound as vocal or instrumental

The final step in the extraction process is to choose and compute a model (linear regression, model trees, knn, locally weighted regression, neural networks, etc.) that combines all features. As an output, EDS creates an executable file, which classifies an audio file passed as argument.

In short, the user needs to 1) create the database, in which each recording is labeled as its correspondent class. 2) write a general pattern for the features and launch the genetic search. The pattern encapsulates the overall procedure of the feature. For example, *!_x(f:a(Signal))* means that the signal is initially converted into the frequency domain, then some operation is applied to get a single value as a result. 3) select the appropriate features. 4) choose a model to combine the features. Although an expert user may drive the system (starting from an initial solution, including heuristics for the genetic search, etc), EDS has a fully automated mode, in which a default pattern is chosen, the most complementary features are selected and all models are computed. This mode is particularly attractive for non expert user, as he/she just needs to be able to create and label the database. That is the mode explored in this paper.

5 Bossa Nova Guitar Chords

Our final goal is to create a guitar accompanier in Brazilian "bossa nova" style. Consequently, our chord recognizer has examples of chords played with nylon guitar. The data was taken from D'accord Guitar Chord Database [10], a guitar midi based chord database. The purpose of using it was the richness of the symbolic information present (chord root, type, set of notes, position, fingers, etc.), which was very useful for labelling the data and validating the results. Each midi chord was rendered into a wav file using Timidity++ [12] and a free nylon guitar patch, and the EDS database was created according to the information found in D'accord Guitar database. Even though a midi-based database may lead to distortions in the results, we judge that the comparison between approaches is still valid.

5.1 Chord Classes

We tested the solutions with some different datasets, reflecting the variety of nuances that chord recognition may show:

AMaj/Min – classifies between major and minor chords, given a fixed root (La). There were 101 recordings, labelled in 2 classes.

Chord Type, fixed root – classifies among major, minor, seventh, minor seventh and diminished chords, given a fixed root (A or C). There were 262 samples, divided in 5 classes,

Chord Recognition – classifies major, minor, seventh, minor seventh and diminished chords, in any root. There were 1885 samples, labelled in 60 classes.

80% of each database is settled on as the training dataset and 20% as the testing dataset.

5.2 Pitch Class Profile

In our implementation of the pitch class profile, frequency to pitch mapping is achieved using the logarithmic characteristic of the equal temperament scale, as illustrated in Fig. 5. The intensity of each pitch is computed by summing the magnitude of all frequency bins that correspond to a particular pitch class. The same computation is applied to a white noise and the result is used to normalize the other PCPs.

$$Pitch = \frac{12 \times \log\left(\frac{|Hz_{bin}|}{440}\right)}{\log(2)}$$

Fig. 6. Frequency to pitch mapping

For the *chord recognition* database, PCPs were rotated, meaning that each PCP was computed 12 times, one time for each possible rotation (for instance, a Bm is equivalent to a Am rotated twice). After the PCP computation, several machine learning algorithms could have been applied. We implemented 2 simple solutions. The first

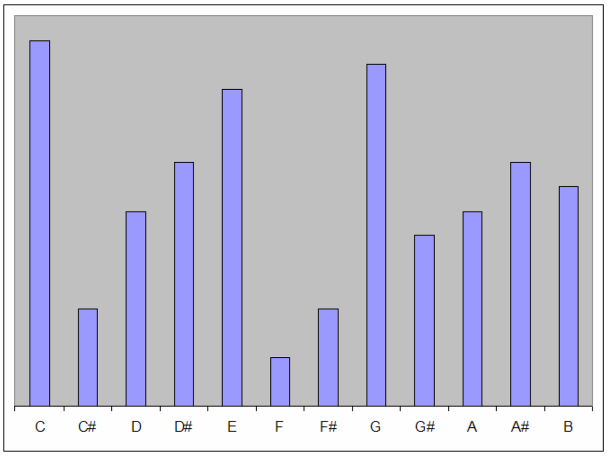


Fig. 7. Example of a template PCP for a C chord class

one calculates a default (template) PCP to each chord class. Then, the PCP of a new example can be matched up to the template PCP, and the most similar one is retrieved as the chord.

The second one uses the k-nearest neighbours algorithm (KNN), with maximum of 3 neighbours. KNNs have been used since the original PCP implementation and have proved to be at least one of the best learning algorithms for this case [3].

5.3 EDS

The same databases were loaded in EDS. We ran a fully automated extraction, keeping all default values. The system generated the descriptor without any help from the user, obtaining the results we call EDS Naïve, because they correspond to the results that a naïve user would achieve.

6 Results and Discussion

The results achieved by us are presented in the table 1. Rows represent the different databases. Columns represent the different learning techniques. The percent values indicate the number of correctly classified instances over the total number of examples in the testing database.

As we can see, EDS gets really close to classical approaches when the root is known, but disappoints when the whole problem is presented. It seems that a combination of low level functions is capable of finding different patterns in the same root, but the current palette of signal processing functions in EDS is not sufficient to generalize harmonic information. Sections 6.1, 6.2 and 6.3 detail the features that were found.

Table 1. Percentage of correctly classified instances for the different databases using the studied approaches

Approach / Database	PCP Template	KNN	EDS
Maj/Min (fixed root)	100%	100%	90.91%
Chord Type (fixed root)	89%	90.62%	87.5%
Chord Recognition	53.85%	63.93%	40.31%

6.1 Case 1: Major/Minor Classifier, Fixed Root

Figure 5 shows the selected features for the *Amaj/min* database. The best model obtained was a KNN of 1 nearest neighbour, equally weighted, absolute error (see [9] for details). The descriptor reached 90.91% of the performance of the best traditional classifier.

```
EDS1: Power (Log10 (Abs (Range (Integration (Square (Mean (FilterBank
(Normalize (Signal), 5.0))))))), -1.0)

EDS2: Power (Log10 (Abs (Range (Sqrt (Bartlett (Mean (FilterBank
(Normalize (Signal), 9.0))))))), -1.0)

EDS3: Sqrt (Range (Integration (Hanning (Square (Mean (Split (Signal,
3736.0)))))))

EDS4: Arcsin (Sqrt (Range (Integration (Mean (Split (Normalize (Sig-
nal), 5862.0))))))

EDS5: Log10 (Variance (Integration (Bartlett (Mean (FilterBank (Nor-
malize (Signal), 5.0))))))

EDS6: Power (Log10 (Abs (Range (Integration (Square (Sum (FilterBank
(Normalize (Signal), 9.0))))))), -1.0)

EDS7: Square (Log10 (Abs (Mean (Normalize (Integration (Normalize
(Signal)))))))

EDS8: Arcsin (Sqrt (Range (Integration (Mean (Split (Normalize (Sig-
nal), 8913.0))))))

EDS9: Power (Log10 (Abs (Range (Sqrt (Bartlett (Mean (FilterBank
(Normalize (Signal), 3.0))))))), -1.0)
```

Fig. 8. Selected features for the *Amaj/min* chord recognizer

6.2 Case 2: Chord Type Recognition, Fixed Root

Figure 6 shows the selected features for the chord type database. The best model obtained was a GMM of 14 gaussians and 500 iterations (see [9] for details). The descriptor reached 96,56% of the performance of the best traditional classifier.

```
EDS1: Log10 (Abs (RHF (Sqrt (Integration (Integration (Normalize
(Signal)))))))

EDS2: Mean (Sum (SplitOverlap (Sum (Bartlett (Split (Signal,
1394.0))), 4451.0, 0.5379660839449434)))

EDS3: Power (Log10 (Abs (RHF (Normalize (Integration (Integration
(Normalize (Signal))))))), 6.0)

EDS4: Power (Log10 (RHF (Signal)), 3.0)

EDS5: Power (Mean (Sum (SplitOverlap (Sum (Bartlett (Split (Signal,
4451.0))), 4451.0, 0.5379660839449434))), 3.0)
```

Fig. 9. Selected features for the Chord Type recognizer

6.3 Case 3: Chord Recognition

Figure 7 shows some of the selected features for the chord recognition database. The best model obtained was a KNN of 4 nearest neighbours, weighted by the inverse of the distance (see [9] for details). The descriptor reached 63,05% of the performance of the best traditional classifier. It is important to notice that 40,31 % is not necessarily a bad result, since we have 60 possible classes. In fact, 27,63% of the wrongly classified instances were due to mistakes between relative majors and minors (e.g; C and Am); 40,78% due to other usual mistakes (e.g. C and C7; C° and Eb°; C and G); only 31,57% were caused by unexpected mistakes. Despite these remarks, the comparative results are significantly worse than the previous ones.

```
EDS1: Square (Log10 (Abs (Sum (SpectralFlatness (Integration (Split
(Signal, 291.0)))))))

EDS4: Power (Log10 (Abs (Iqr (SpectralFlatness (Integration (Split
(Signal, 424.0)))))), -1.0)

EDS9: Sum (SpectralRolloff (Integration (Hamming (Split (Signal,
4525.0)))))

EDS10: Power (Log10 (Abs (Median (SpectralFlatness (Integration
(SplitOverlap (Signal, 5638.0, 0.7366433546185794)))))), -1.0)

EDS12: Log10 (Sum (MelBands (Normalize (Signal), 7.0)))

EDS13: Power (Median (Normalize (Signal)), 5.0)

EDS14: Rms (Range (Hann (Split (Signal, 9336.0))))

EDS15: Power (Median (Median (Split (Sqrt (Iqr (Hamming (Split (Sig-
nal, 2558.0)))), 4352.0))), 1.5)

EDS17: Power (HFC (Power (Correlation (Normalize (Signal), Signal),
4.0)), -2.0)

EDS18: Square (Log10 (Variance (Square (Range (Mfcc (Square (Hamming
(Split (Signal, 9415.0))), 2.0))))))

EDS19: Variance (Abs (Median (Hann (FilterBank (Peaks (Normalize
(Signal)), 5.0)))))

EDS21: MaxPos (Sqrt (Normalize (Signal)))

EDS22: Power (Log10 (Abs (Iqr (SpectralFlatness (Integration (Split
(Signal, 4542.0)))))), -1.0)
```

Fig. 10. Some of the selected features for the chord recognizer

6.4 Other Cases

We also compared the three approaches on other databases, as we can see in the table 2. *MajMinA* is the major/minor classifier, root fixed to A. *ChordA* is the chord type recognizer, root fixed to A. *ChordC* is the chord type recognizer, root fixed to C. *RealChordC* is the same chord type recognizer in C, but the testing dataset is composed by real audio recordings (samples of less than 1 second of chords played in a nylon guitar), instead of midi rendered audio. Curiously, in this case, the EDS solution worked better than the traditional one (probably due to an alteration in tuning in the recorded audio). *Chord* is the chord recognition database. SmallChord is a smaller dataset (300 examples) for the same problem. Notice that in this case EDS outperformed KNN and PCP Template. In fact, the EDS solution does not improve very much when passing from 300 to 1885 examples (from 38,64% to 40,31%), while the

KNN solution goes from 44% to 63,93%. Finally, *RealChord* has the same training set from the *Chord* database, but is tested with real recorded audio.

The results from these databases confirm the trend of the previous scenario. The reading of the results indicates that the effectiveness of the EDS fully automated descriptor extraction depends on the domain it is applied to. Even admitting that EDS (in its current state) is only partially suited to non expert users, we must take into account that EDS currently uses a limited palette of signal processing functions, which is being progressively enhanced. Since EDS didn't have any information about tonal harmony, it was already expected that it would not reach the best results. Even though, the results obtained by the chord recognizer with a fixed root show the power of the tool.

Table 2. Comparison between the performance of the EDS and the best traditional classifier for a larger group of databases. Comparative performance = EDS performance / traditional technique performance.

DB NAME	Comparative Performance
MajMinA	90,91%
ChordA	94,38%
ChordC	96,56%
Chord	63,05%
SmallChord	87,82%
RealChordC	116,66%
RealChord	55,16%

7 Conclusion and Future Works

In this paper we compared the performance of a standard chord recognition technique and the EDS approach. The chord recognition was specifically related to nylon guitar samples, since we intend to apply the solution to a Brazilian style guitar accompanier. The standard technique was the Pitch Class Profiles, in which frequency intensities are mapped to the twelve semitone pitch classes, and then uses KNN classification to chord templates. EDS is an automatic descriptor extractor system that can be employed even if the user does not have knowledge about signal processing. It was operated in a completely naïve way so that the solution and the results would be similar to those obtained by a non expert user.

The statistical results reveal a slight deficit of EDS for a fixed root, and a greater gap when the root is not known a priori, showing its dependency on primary operators. An initial improvement is logically the increase of the palette of functions. Currently, we are implementing tonal harmony operators such as chroma and pitchBands, which we suppose will provide much better results. Additionally, as the genetic search in EDS is indeed an optimisation algorithm, if the user starts from a good solution, it will be expected that the algorithm makes it even better. The user can also guide the function generation process, via more specific patterns and heuristics.

With these actions, we intend to perform the second part of the comparison we started in this paper – between the traditional techniques and EDS operated by a signal processing expert.

Acknowledgements

We would like to thank all the team at Sony CSL Paris, particularly Anthony Beurivé, Jean-Julien Aucouturier and Aymeric Zils for their support and assistance with EDS; and a special thanks to Tristan Jehan for his help in the conception and implementation of the algorithms.

References

1. Pachet, F. and Zils, A. "Automatic Extraction of Music Descriptors from Acoustic Signals", Proceedings of Fifth International Conference on Music Information Retrieval (ISMIR04), Barcelona, 2004.
2. Zils, A. & Pachet, F. "Extracting Automatically the Perceived Intensity of Music Titles", Proceedings of the 6th COST-G6 Conference on Digital Audio Effects (DAFX03), 2003.
3. Gómez, E. and Herrera, P. "Estimating the tonality of polyphonic audio files: cognitive versus machine learning modelling strategies", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona, 2004.
4. Sheh, A. and Ellis, D. "Chord Segmentation and Recognition using EM-Trained Hidden Markov Models", Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR03), Baltimore, USA, 2003.
5. Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T. and Okuno, H. "Automatic chord transcription with concurrent recognition of chord symbols and boundaries", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona, 2004.
6. Fujishima, T. "Real-time chord recognition of musical sound: a system using Common Lisp Music", Proceedings of International Computer Music Conference (ICMC99), Beijing, 1999.
7. Bartsch, M. A. and Wakefield, G. H. "To catch a chorus: Using chromabased representation for audio thumbnailing", Proceedings of International. Workshop on Applications of Signal Processing to Audio and Acoustics, Mohonk, USA, 2001.
8. Pardo, B., Birmingham, W. P. "The Chordal Analysis of Tonal Music", The University of Michigan, Department of Electrical Engineering and Computer Science Technical Report CSE-TR-439-01, 2001.
9. Mitchell, T. "Machine Learning", The McGraw-Hill Companies, Inc. 1997.
10. Cabral, G., Zanforlin, I., Santana, H., Lima, R., & Ramalho, G. "D'accord Guitar: An Innovative Guitar Performance System", in Proceedings of Journées d'Informatique Musicale (JIM01), Bourges, 2001.
11. Koza, J. R. "Genetic Programming: on the programming of computers by means of natural selection", Cambridge, USA, The MIT Press.
12. Gómez, E. Herrera, P. "Automatic Extraction of Tonal Metadata from Polyphonic Audio Recordings", Proceedings of 25th International AES Conference, London, 2004.
13. Website: http:// timidity.sourceforge.net/

Improving Prototypical Artist Detection by Penalizing Exorbitant Popularity

Markus Schedl[1,2], Peter Knees[1], and Gerhard Widmer[1,2]

[1] Department of Computational Perception,
Johannes Kepler University (JKU),
A-4040 Linz, Austria
[2] Austrian Research Institute for Artificial Intelligence (FAI),
A-1010 Vienna, Austria
{markus.schedl, peter.knees, gerhard.widmer}@jku.at

Abstract. Discovering artists that can be considered as prototypes for particular genres or styles of music is a challenging and interesting task. Based on preliminary work, we elaborate an improved approach to rank artists according to their prototypicality. To calculate such a ranking, we use asymmetric similarity matrices obtained via co-occurrence analysis of artist names on web pages. In order to avoid distortions of the ranking due to ambiguous artist names, e.g. bands whose name equal common speech words (like "Kiss" or "Bush"), we introduce a penalization function. Our approach is demonstrated on a data set containing 224 artists from 14 genres.

1 Introduction and Related Work

Prototypical artist detection provides valuable information for a wide range of applications related to music information retrieval. For example, music information systems like the All Music Guide[1] as well as online music stores, e.g. Amazon[2], could benefit considerably. For instance, information on prototypes could be exploited to support their users in finding music more efficiently. Furthermore, prototypical artists are very useful for visualizing music repositories since they are usually well-known. Thus, they can serve as reference points to discover similar but less known artists (for more details, see [4]).

To obtain an estimate for the prototypicality of artists, we exploit information on co-occurrences of artist names on web pages. We already showed that web-based co-occurrence analysis can be used successfully for artist similarity measurement and artist-to-genre classification [3]. In this paper, we will use the (approximate) page counts of Google requests for artists to estimate conditional probabilities for an artist to be found on web pages of other artists. These probabilities give an asymmetric similarity matrix which is used for the calculation

[1] *http://www.allmusic.com*
[2] *http://www.amazon.com*

of a prototypicality ranking (cf. [3]). By introducing a penalty term in the definition of the ranking function, we overcome the emerging problem of extreme high rankings for artists with common speech names.

The remainder of this paper is organized as follows. In Section 2, we briefly review our already published approach and propose a method to remedy the shortcoming of extreme prototypicality for artists that allegedly occur on many web pages. Furthermore, we demonstrate the improvements in the method on the basis of selected results. Finally, in Section 3, we summarize the work and draw conclusions.

2 Prototypical Artist Detection

The obtain the information used for Prototypical Artist Detection, at first we perform a co-occurrence analysis step. Given a list of artist names, we use Google to estimate the number of web pages containing each artist and each pair of artists. To this end, the only information we need is the page count returned by Google. This raises performance and limits web traffic. Based on the page counts, we then use relative frequencies to calculate a conditional probability matrix. Given two events a_i (artist with index i is mentioned on web page) and a_j (artist with index j is mentioned on web page), we estimate the conditional probability p_{ij} (the probability for artist j to be found on a web page that is known to contain artist i). The formula for calculation can be found in detail in [3]. From this, we obtain a similarity matrix that is obviously not symmetric.

2.1 Prototype Detection Using Backlink/Forward Link Ratios

We regard the prototypicality of a music artist as being strongly related to how often music-related web pages refer to the artist. The used method to infer prototypicality is based on an idea similar to the PageRank mechanism used by Google where *backlinks* and *forward links* of a web page are used to measure relevancy [1]. In our approach we call any co-occurrence of artist a and artist b (unequal to a) on a web page that is known to contain artist b a *backlink* of a (from b). A *forward link* of an artist of interest a to another artist b, in contrast, is given by any occurrence of artist b on a web page which is known to mention artist a.

For each artist a_i, we count for how many of the artists a_j from the same genre as a_i the number of backlinks of a_i exceeds the number of forward links (bl), and also for how many artists how often the number of forward links of a_i is higher than the number of backlinks (fl). Then, we calculate the ratio bl/fl. The higher this ratio, the higher the prototypicality of a_i for the respective genre. A more formal definition can be found in [4].

2.2 Downranking Artists by Penalizing Exorbitant Popularity

As results in [4] show, artist names which are also used in everyday speech are always top-ranked, for example, *Kiss* from the genre Heavy Metal/Hard Rock,

Table 1. Artist ranking according to corrected prototypicality for some genres of the test set. Moreover, the *ranking value*, the original *backlink/forward link (bl/fl) ratio*, and the *penalty* is shown for every artist.

Pop				*Heavy Metal/Hard Rock*				*Alternative Rock/Indie*			
artist ranking	rank	bl/fl	pen	artist ranking	rank	bl/fl	pen	artist ranking	rank	bl/fl	pen
Britney Spears	0.20	13:2	0.22	Metallica	0.44	14:1	0.25	Radiohead	0.18	11:4	0.29
Madonna	0.20	14:1	0.17	Slayer	0.38	13:2	0.29	Weezer	0.17	8:7	0.41
Janet Jackson	0.19	10:5	0.33	Iron Maiden	0.33	11:4	0.39	Beck	0.16	12:3	0.23
Avril Lavigne	0.17	10:5	0.32	AC/DC	0.30	12:3	0.32	Pearl Jam	0.16	9:6	0.36
Jennifer Lopez	0.14	9:6	0.33	Anthrax	0.24	10:5	0.38	Nirvana	0.14	13:2	0.18
Michael Jackson	0.14	12:3	0.22	Black Sabbath	0.19	9:6	0.38	Smashing Pumpkins	0.13	7:8	0.40
Christina Aguilera	0.12	8:7	0.35	Def Leppard	0.17	8:7	0.41	Sonic Youth	0.13	10:5	0.27
Robbie Williams	0.10	7:8	0.36	Kiss	0.16	15:0	0.10	Hole	0.12	14:1	0.13
ABBA	0.09	6:9	0.38	Deep Purple	0.14	7:8	0.42	Depeche Mode	0.10	6:9	0.41
N'Sync	0.08	4:11	0.50	Megadeth	0.11	6:9	0.43	Foo Fighters	0.09	5:10	0.44
Prince	0.06	15:0	0.06	Pantera	0.09	5:10	0.44	Belle and Sebastian	0.08	3:12	0.57
Justin Timberlake	0.05	4:11	0.40	Alice Cooper	0.07	4:11	0.45	Alice in Chains	0.06	3:12	0.49
Spice Girls	0.05	3:12	0.48	Judas Priest	0.05	3:12	0.46	The Smiths	0.05	3:12	0.46
Shakira	0.05	3:12	0.46	Sepultura	0.04	2:13	0.53	Jane's Addiction	0.02	1:14	0.58
O-Town	0.03	1:14	0.64	Skid Row	0.02	1:14	0.54	Bush	0.00	15:0	0.00
Nelly Furtado	0.02	1:14	0.50	Queensryche	0.00	0:15	0.55	Echo and the Bunnymen	0.00	0:15	0.69

Bush, *Hole*, and *Nirvana* from Alternative Rock/Indie, or *Prince* and *Madonna* from Pop. The reason for this is that such words frequently occur on web pages and therefore produce a lot of backlinks for artists with the such names. This kind of misleading co-occurrences are a shortcoming of web-based information retrieval methods and could also distort the prototypicality ranking.

To avoid such distortions, we propose a mechanism that basically pursues the idea of the commonly used information retrieval approach $tf \times idf$ (term frequency×inverse document frequency) [2]. In this approach, importance of a term is higher, if the term occurs frequently (high tf). On the other hand, a term is penalized, if it occurs in many documents and thus does not contain much relevant information (high df leads to low idf).

In our approach, we adapt this principle to penalize the prototypicality of an artist, if it high over all genres (following the naming scheme of $tf \times idf$, we could call our approach $gp \times iop$ for *genre prototypicality×inverse overall prototypicality*). This is reasonable, since even for very popular and important artists it is unlikely to be a prototype for all types of music and all artists. Furthermore, common speech words appear on artists' web pages independently of their genre. Taking a closer look of the overall bl/fl ratios from [4] supports this consideration. On the set of 224 artists, those artists with common speech word names yield by far the highest overall bl/fl ratios, i.e. *Bush* (223/0), *Prince* (222/1), *Kiss* (221/2), *Madonna* (220/3), and *Nirvana* (218/5). Thus, using the information from the global prototypicality, we suggest ranking of artists according to the value

$$prot(a) = \frac{bl_{genre}(a)}{fl_{genre}(a) + 1} \cdot penalty(a)^2, \tag{1}$$

where

$$penalty(a) = norm(log\frac{fl_{global}(a)}{bl_{global}(a) + 1}), \tag{2}$$

and *norm* is a function that shifts all values in the positive range by subtracting the smallest (non negative infinite) value, replaces infinite numbers by 0, and normalizes the values by division by the maximum value (in the order mentioned).

To demonstrate the effects of this revised function, Table xx shows the newly obtained rankings for genres containing artists with common speech word names, as well as one ranking for a genre without such artists that remained almost unmodified. Concerning evaluation we refer at the results published in [4]. For genres without ambiguous artist's names results (and thus correlation with the ranking obtained through Google queries) remain constant in principle. For the other genres, correlation decreases. This is no negative result, since it is the intention of our approach to overcome the susceptibility of web-based approaches for overrating of common word names.

3 Conclusions and Future Work

We presented an approach for automatic detection of prototypical artists. To this end, we use asymmetric similarity matrices gained by co-occurrence analysis of artist names on web pages. Based on these similarity matrices, we estimate a prototypicality ranking for the artists using *backlink/forward link ratios*. To overcome the problem of unjustified high rankings for artists whose name equal common speech words, we apply a penalization function that weights the local bl/fl ratios (per genre) with a penalty. This penalty is calculated using global bl/fl ratios (computed on the complete artist set).

We demonstrated our approach on a test collection containing 224 artists of 14 genres. It was shown that the introduction of a penalization function corrects the ranking results obtained by the simple bl/fl prototypicality ranking.

As for future work, it is planned to evaluate our approach on a larger artist set containing 953 artists from 15 genres. This would be highly interesting since a shortcoming of the test collection used here is that most of its artists are quite popular and typical of their genre. Furthermore, we are currently elaborating on heuristics to minimize the computational complexity of the co-occurrence analysis and we hope to gain interesting insights in the near future.

Acknowledgments

This research is supported by the Austrian Fonds zur Frderung der Wissenschaftlichen Forschung (FWF) under project numbers L112-N04 and Y99-START, and by the EU 6th FP project SIMAC (project number 507142).

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Education, Science, and Culture and by the Austrian Federal Ministry for Transport, Innovation, and Technology.

References

1. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. In *Proc. of the Annual Meeting of the American Society for Information Science*, pages 161–172, January 1998.
2. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
3. M. Schedl, P. Knees, and G. Widmer. A Web-Based Approach to Assessing Artist Similarity using Co-Occurrences. In *Proc. of the 4th International Workshop on Content-Based Multimedia Indexing*, Riga, Latvia, June 2005.
4. M. Schedl, P. Knees, and G. Widmer. Discovering and Visualizing Prototypical Artists by Web-based Co-Occurrence Analysis. In *Proc. of the 6th International Conference on Music Information Retrieval*, London, UK, September 2005.

Music Analysis and Modeling Through Petri Nets

Adriano Baratè, Goffredo Haus, and Luca A. Ludovico

LIM-DICO University of Milan,
Via Comelico, 39,
20135 Milano, Italy
{barate, haus, ludovico}@dico.unimi.it

Abstract. Petri Nets are a formal tool for studying systems that are concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. They were used in a number of real-world simulations and scientific problems, but seldom considered an effective means to describe and/or generate music. The purpose of this paper is demonstrating that Petri Nets (enriched with some peculiar extensions) can well represent the results of a musicological analysis process.

1 Introduction

This paper represents the results recently obtained at LIM (Musical Informatics Laboratory, State University of Milan) in the area of Music Petri Nets (Music PNs). This mathematical formalism, shortly introduced and defined in the following sections, can be applied to music field according to different meanings. Roughly, we can recognize two possible categories of applications: analysis and composition. However, the former and the latter aspects cannot be considered completely independent. In fact, PN-oriented analysis would provide poor results, if not aimed at the comprehension of the original composition or even at the generation of a new music piece that shares some common features with the one previously analyzed. Moreover, PN-based composition itself would produce insignificant results, if not supported by a deep comprehension of the underlying structures, which involves – after all – an analysis process. In this paper, we will concentrate on the analytical possibilities and limitations related to the application of Petri Nets to music field. The possible consequences in music composition will not be explored, and will represent one of the subjects of our future work.

When applied to music analysis activities, the adoption of Petri Nets should not influence the approach of the researcher. In fact, Petri Nets should be thought as a way (one of many possible ways) to express the results of the analytical process. From our perspective, this formalism cannot limit or influence the analysis, which is an activity that obviously precedes the representation of its results.

We said that Petri Nets should not constitute a limitation, but this is not sufficient: we want to demonstrate that they are useful and effective. We will show that Petri Nets are a promising tool to represent and read music analysis results. Petri Nets were born to describe concurrent, asynchronous, and parallel processes, and these characteristics can be found in music as well. Other characteristics, such as non-determinism, can be useful for music composition through Petri Nets, but are not very significant for analysis.

In the following discussion, no constraints will be imposed about music to analyze: music works can belong to different genres, and come from different cultures, geographical areas and historical periods. On the contrary, we will underline the adequacy of Petri Nets formalism according to different degrees of abstraction in analysis, which will be the subject of next section.

2 Music Analysis and Grouping Structures

In this section we introduce the formal concept of *grouping structure*, as defined in [6]. A *group* can be constituted by any contiguous sequence of pitch events, undetermined beats or rests. Only contiguous sequences can constitute a group. A group can contain smaller groups, and in this case the subgroups must be completely contained in the former. Finally, if a group contain at least a smaller group, it should be possible to partition it exhaustively in smaller groups. These conditions define a strict, non-overlapping, recursive hierarchy, and constitute a set of grouping well-formedness rules. Intentionally, we don't introduce at the moment a set of grouping preference rules.

In this context, a music piece as well as a single note can constitute a group. Besides, the identification of grouping structures allows extracting from the score music objects such as episodes, themes, rhythmic patterns, or harmonic cadences.

Fig. 1 illustrates in a hierarchical fashion some possible groupings for a melody.[1] To reflect hierarchies, groups are represented by slurs placed beneath the music notation.

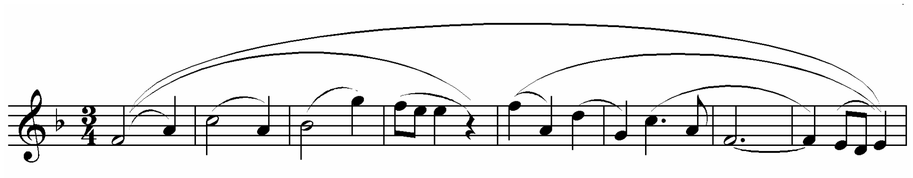


Fig. 1. Examples of grouping structures

Considering only 8 measures imposes serious limitations to the reachable degree of analysis: it doesn't allow the segmentation in episodes of the whole first movement, or the identification of recurrences of the music object represented in Fig. 1. Nevertheless, at least three categories of grouping structures can be identified. The most comprehensive structure (i.e. the largest group) embraces a whole period, whereas the two subsumed groupings reflect the subdivision of the 8-measures period in two 4-measures phrases. More interesting, the third proposed grouping structure tries to highlight relationships among smaller music objects: for

[1] W. A. Mozart, *Sonata in F major KV 332 (300k) – Allegro (I movement)*, bars 1-8 [8]. All the music examples in this paper have been extracted from this piece.

instance, the first 3 groups (measure 1, 2, and 3 respectively) present rhythmic similarity; group 5, 6, and 7 (in measures 5, 6, 7, and 8) have a similar melodic behaviour. Of course, other segmentations could be realized, by using different criteria for grouping structure identification.

It's worth to note that our concept of grouping structure can embrace also vertical slices of a music piece, which allows harmonic analysis and the segmentation of a piece in episodes.

In the simple example shown in Fig. 1, we considered only a small portion of a voice. Of course, the identification of grouping structures should be extended to the whole piece or even to a set of pieces that constitute an overall music work (e.g. the movements of a sonata or the episodes of a symphonic poem). According to our previous definitions, the analytical process for a music work can be thought – in general terms – as the identification of grouping structures together with their relationships inside the piece. Intentionally, this statement is quite vague. In fact, our purpose is specializing such definition in a number of different ways (creating different degrees of abstraction in music analysis) in order to demonstrate the adequacy of Petri Nets to the different cases.

From the musicologist's perspective, identifying grouping structures is the first step to establish relationships among them, highlighting similarities and differences. For instance, the music form known as *sonata* is characterized by the presence of two contrasting themes (the principal and the secondary theme), that are reproposed in a literal or slightly varied form during the piece, according to given rules and to the inspiration of the composer. After identifying those grouping structures, it is possible to show the alternations and recurrences of themes and other transition music objects: in other words, the analysis focuses also the relationships among music objects.

The analytical process can be conducted at different degrees of abstraction, addressing movements in a complex composition, macro-episodes in a piece, themes in an episode or even atomic music events in an elementary music object. In this work we will show how music structures can be highlighted and also processed by means of a more abstract kind of representation than the staff one. Petri Nets, a formalism that will be soon described, are able to represent effectively the results of music analysis depending on the degree of abstraction we want to introduce in the process.

3 An Introduction to Petri Nets

A Petri Net (PN) is an abstract and formal model to represent the dynamic behaviour of a system with asynchronous and concurrent activities ([4], [9], [10]).

PNs consist in a set of basic objects: places, transitions and arcs, whose graphical representations are circles, rectangles, and oriented lines respectively. Places and transitions are also called nodes.

In Fig. 2, an example of an elementary PN is shown. P1, P2, P3, P4 are places, T1, T2, T3 are transitions, and the oriented lines represent arcs. The number associated to arcs is called *arc weight*.

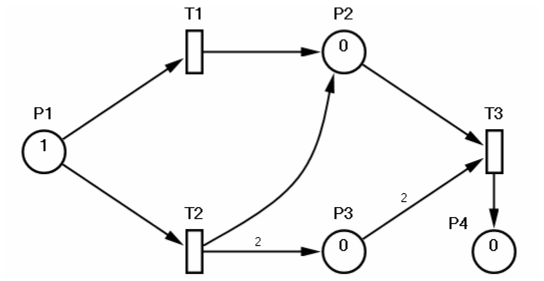


Fig. 2. An example of Petri Net

When building a PN, some constraints about elements layout must be respected. An arc can connect only nodes of different kind, i.e. a place to a transition or a transition to a place. However, two or more arcs having the same orientation can connect two nodes. From a graphical point of view, such a behavior can be summarized using the concept of arc weight, which represents the multiplicity of the arc.

A key concept of PNs is the idea of *marking*, realized using tokens. Any place can hold a certain number of tokens, usually represented by little black circles; however, in a computer-oriented description, a better formalism is adopted: a numerical value inside the place indicates the total number of tokens in that place at a given time. This is the meaning of the value inside the places in Fig. 2.

Tokens let PNs evolve and self-modify. They can be transferred from place to place according to policies known as firing rules. The dynamic evolution of a PN is determined by the following rules:

- When all the incoming places of a transition present a number of tokens greater or equal to the weights of the corresponding incoming arcs, the transition is enabled.
- When a transition is enabled, the fire drops from the incoming places a number of tokens equal to the weights of the incoming arcs and adds to each outgoing place a number of tokens equal to the weights of the corresponding outgoing arc.

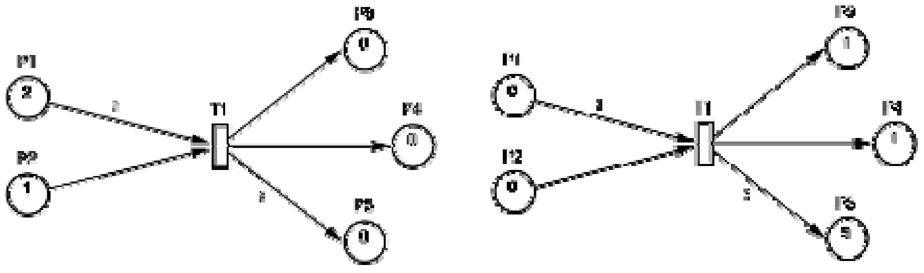


Fig. 3. Evolution of a Petri Net

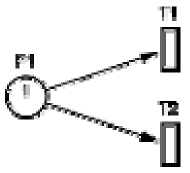


Fig. 4. Alternative transitions

When a transition is enabled, its fire is not automatic. For instance, looking at Fig. 4, T1 and T2 are both enabled, but there is only one token in input, so they both cannot fire. In such a case, we call them *alternative transitions*, and only one of the two transitions will fire. This is a kind of non-determinism.

A net execution is formed by subsequent transition fires, and terminates when no more transitions are enabled.

3.1 Extensions

In order to use PNs as an effective tool to describe music, we have to define further extensions to the given definition. Not all the extensions presented here are original: many of them are already in use in PNs general applications.

Capacity. The capacity is a property of places, and indicates the maximum number of housed tokens. This attribute creates a new condition for transition enabling: a transition cannot be enabled if the marking of at least one output place would become greater than its capacity after the fire of the transition.

The introduction of the capacity concept adds a new type of non-determinism (see also Fig. 4), called *conflict.* A conflict occurs when two (or more) transitions are enabled, but the fire of one transition prevents the fire of the other one(s), according to their capacities (see Fig. 5).

In our graphical representations, the upper value inside a place represents its present marking (number of tokens), whereas the lower one indicates its capacity.

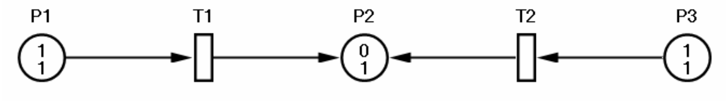


Fig. 5. An example of conflict

Refinement. The theory of morphisms is complex, and this is not the place for an exhaustive treatment. However, we introduce the concept of *refinement*, a simple morphism used to describe complex PNs in terms of simpler ones, in a hierarchical way. A refinement, called *subnet* in this context, represents an entire PN that replaces a node. For instance, in Fig. 6, the place P2 subtends the PN on the right; the expansion of such node would generate the lower global PN. A subnet must have an input node and an output node of the same type of the refined node.

Temporization. As stated above, PNs are a good way to describe concurrent processes together with their synchronization. From a temporal perspective, when a transition is enabled, the duration of the fire is supposed to be null, so a PN execution can

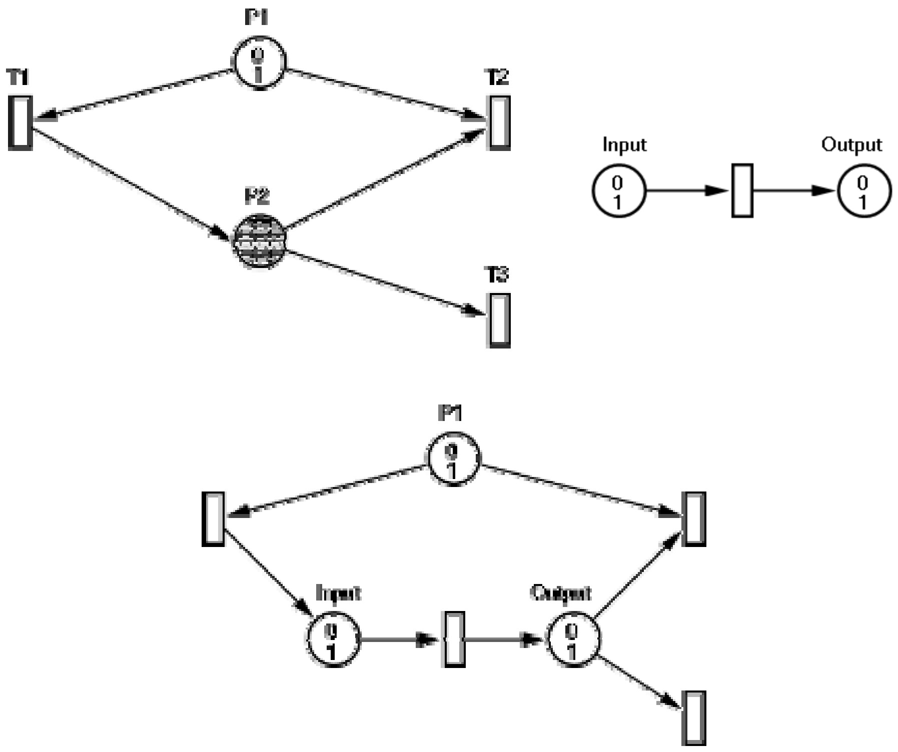


Fig. 6. Refinements and subnets

be considered instantaneous. In Music PNs, we will necessarily introduce the concept of temporization, as music processes are temporized and music objects present a temporal dimension.

Probabilistic Weight. In order to face conflicting or alternative situations, we have to introduce another extension: the *probabilistic weight* of arcs. When many transitions are enabled, the choice depends on the probabilistic weight of the arcs involved, in relation with the total sum of their weights.

For example, let us consider a PN with 3 arcs: A_1 (probabilistic weight $W_1 = 5$), A_2 ($W_2 = 10$), and A_3 ($W_3 = 300$). If at a given time t_1 the choice is between all the three arcs, A_1 shall have a probability of 5/315 (1.6%) to fire, A_2 a probability of 10/315 (3.2%), and A_3 a probability of 300/315 (95.2%). At the time $t_2 > t_1$, let only A_1 and A_2 be enabled: their new probabilities will be 5/15 (33.3%) and 10/15 (66.7%) respectively.

A particular situation occurs when an arc has a probabilistic weight equal to 0. In this case, the associated transition will fire only if there are no other alternative or conflicting arcs with greater probabilistic weight.

Probabilistic weight will be graphically represented by a numeric value over the arc, in square brackets.

3.2 Music PNs

In Music PNs, we can associate music objects to places. According to the definition in [2], a *music object* may be anything that could have a music meaning and that could be thought as an entity, either simple or complex, either abstract or detailed. Such entity will present some relationship with other music objects. When a place containing an object receives a token, the music object is executed. Fig. 7 shows two simple music objects that will be helpful to understand the following examples about transitions' behavior.

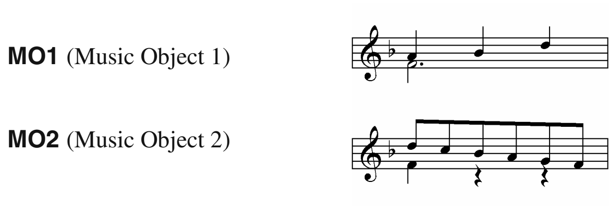


Fig. 7. Two music objects: MO1 and MO2

Transitions play an important role: they determine – together with tokens – the evolution of the net. In our extension of PNs, namely Music Petri Nets, two categories can be found: transitions *with* and *without* associated music operators. We can consider the latter category as transitions having a null operator associated.

When there is no music operator associated, transitions present a simpler behavior: they are only devoted to net evolution. Their role is dropping tokens from input places and adding them accordingly to output places. As noted before, when a token arrives at a place, if the place has an associated music object, this object is played. In Music PNs, the temporization is performed accordingly to the durations of the music objects (eventually) associated to the places. When a place receives a token from an incoming transition, the (eventually) associated music fragment is executed, and the token cannot be used until such execution is completed.

An example is provided in Fig. 8, where MO1 is associated to the left place and MO2 is associated to the right one. The first measure comes from the execution of MO1, which took place when the token arrived at the left place. Then, the token is moved to the right place, originating the subsequent execution of the second measure. The overall result is noted in the score.

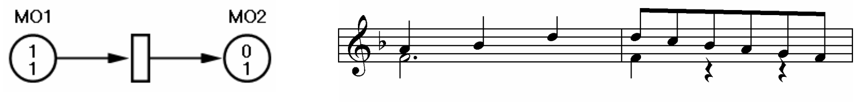


Fig. 8. The *sequence* structure

Even by using transitions *without* music operators, it is possible to create peculiar structures. Fig. 9 provides five simple nets in order to illustrate respectively a fusion (two objects generating one object), a split (an object generating two objects), an alternative (a non-deterministic choice between two objects), and a joint structure (a logical connection between two objects).

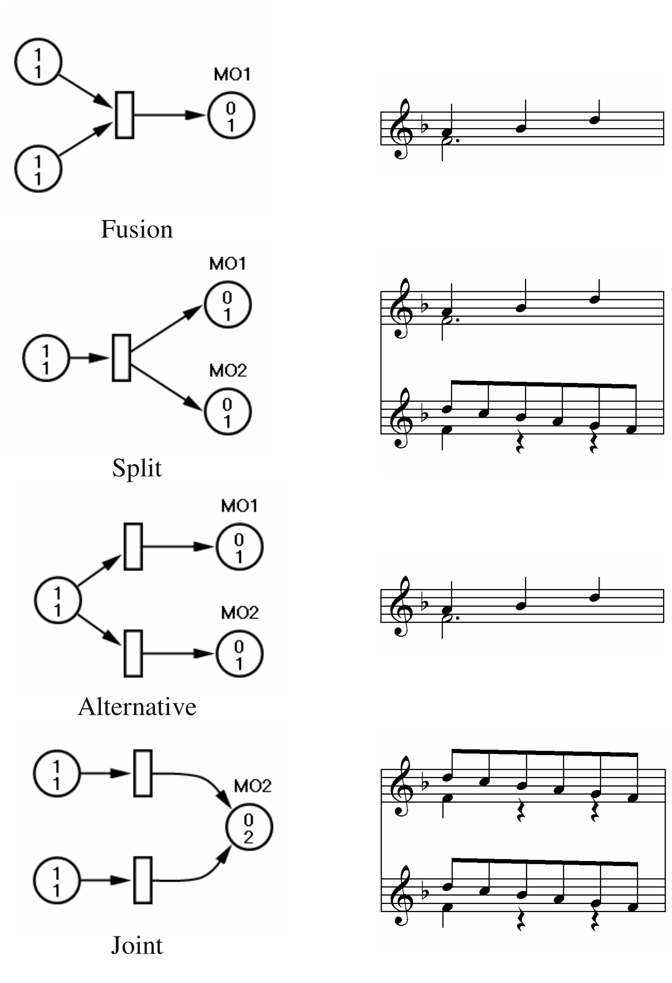


Fig. 9. Some PN structures and the corresponding executions

In the aforementioned approach, we said that transitions might have also an associated music operator. The purpose of music operators is providing changes to input objects (i.e. objects coming from input places), and passing the transformed objects to output places. Typical operators associated to transitions reflect common music operators, such as inversion, retrogradation, and transposition. The behavior of the last operator is shown in Fig. 10.

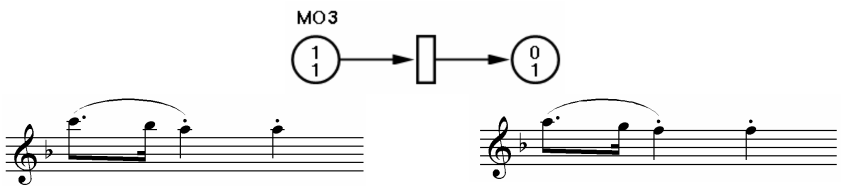


Fig. 10. Music Object 3 before (left) and after (right) the transposition

4 PN-Oriented Analysis and Grouping Structures

The main questions this paper wants to answer are the following: Are PNs a good formalism to collect the results coming from an analytical process, at various levels of detail and abstraction? And are PNs a valid tool to provide a new way to read music structures and to create relationships among music objects? These questions were partially answered by previous works and papers, such as [2].

Apparently, the applications of Petri Nets to music analysis lead to contradictory results. If [3] demonstrates a great efficacy in describing the structure of Ravel's *Bolero* through Petri Nets, on the other hand [1] points out some limitations of this approach, on the base of opportune counterexamples.

An accurate choice of the piece to be analyzed could generate excellent results: this is the case of music works constituted by a few music objects or having a very simple structure. For instance, the music form known as *canon*, based on the (almost) slavish repetition of the same music objects in different voices at different instants, can be represented in a very compact way by using Petri Nets. Provided that music objects are suitably identified, most counterpoint production (including baroque and serial music) originates very compact PN-based descriptions. Another field of application is music with a very simple structure, such as pop songs or deliberately trivial pieces. Ravel's *Bolero* belongs to the latter category: its structure is intentionally simple and repetitive. Probably it would be very difficult to describe a romantic piece or a jazz improvisation by Petri Nets.

In our opinion, PNs can be more or less appropriate to provide a readable and compact description of analysis results depending on the level of detail the analysis wants to reach. This statement justifies the contradictory results obtained within the same research group at LIM when considering different music pieces.

Thus, Petri Nets are more or less efficient and effective depending on the possibility to identify a few music objects and simple relationships among them. According to the aforementioned definition of music object, this concept can embrace whole episodes of a music work as well as single atomic events. As noted before, the identification of music objects is strictly related to the grouping structures we chose for analysis.

Next section will take into consideration the first movement of a sonata by W.A. Mozart. Intentionally, this case study takes an intermediate place between a strongly structured piece and a completely unstructured one: the score can not be described as a sequence of few repetitive music objects (like a canon or fugue), however musicologists agree about the presence of characteristic grouping structures (principal theme, secondary theme, middle themes) and about its macroscopic segmentation (exposition, development, repeat, and coda). This case study will constitute a serious benchmark for Petri Nets application in music analysis.

5 A Case Study: 1st Movement of a Sonata by W.A. Mozart

On the base of E. Surian's text [11], we have analyzed the 1st movement of piano *Sonata KV 332* by W.A. Mozart. The purpose of our analysis is the application of PNs to the structure of the music piece, at different degrees of abstraction.

A typical sonata-form movement consists of a two-part tonal structure, articulated in three main sections [7]. In this piece, the optional introduction and coda are not present.

The first section, called *exposition*, divides into a "first group" in the tonic and, after transitional material, a "second group" in another key. The piece we considered is in major key, so the second group - according sonata form's rules - is presented in the dominant degree. Both groups may include (and in this case actually include) a number of different themes. The first group of themes includes the principal one, namely the *main theme*; the second group of themes introduces the *secondary theme*, together with other thematic material.

The second part of sonata structure is represented by the *development*, which elaborates material from the exposition in a variety of ways, moving through a number of keys. It also prepares the return to the main theme and to the tonic key which begins the following part.

Finally, the third section is named *recapitulation*, and restates the themes of the exposition, usually in the same order. The second group is now heard in the tonic.

Interesting differences occur in the transitional measures, namely the part between the first and the second group of themes: in fact, in the exposition the two groups are in the tonic and in the dominant degree respectively, whereas in the recapitulation they both are in the tonic degree.

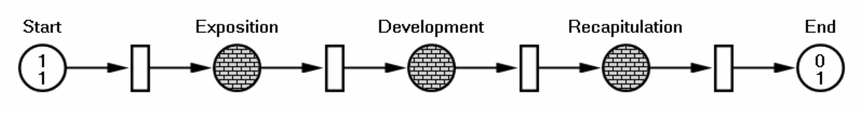
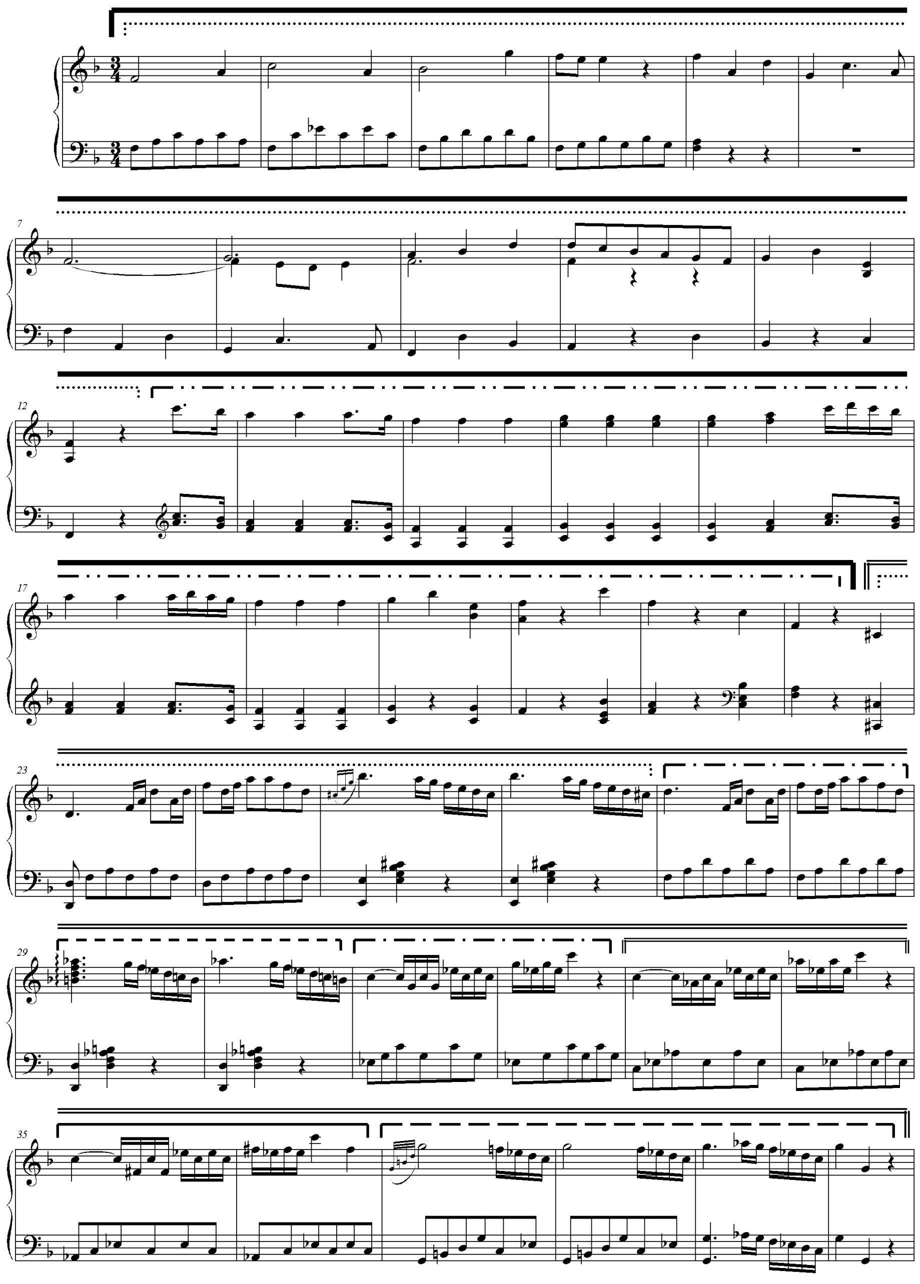


Fig. 11. A high-level representation of sonata form

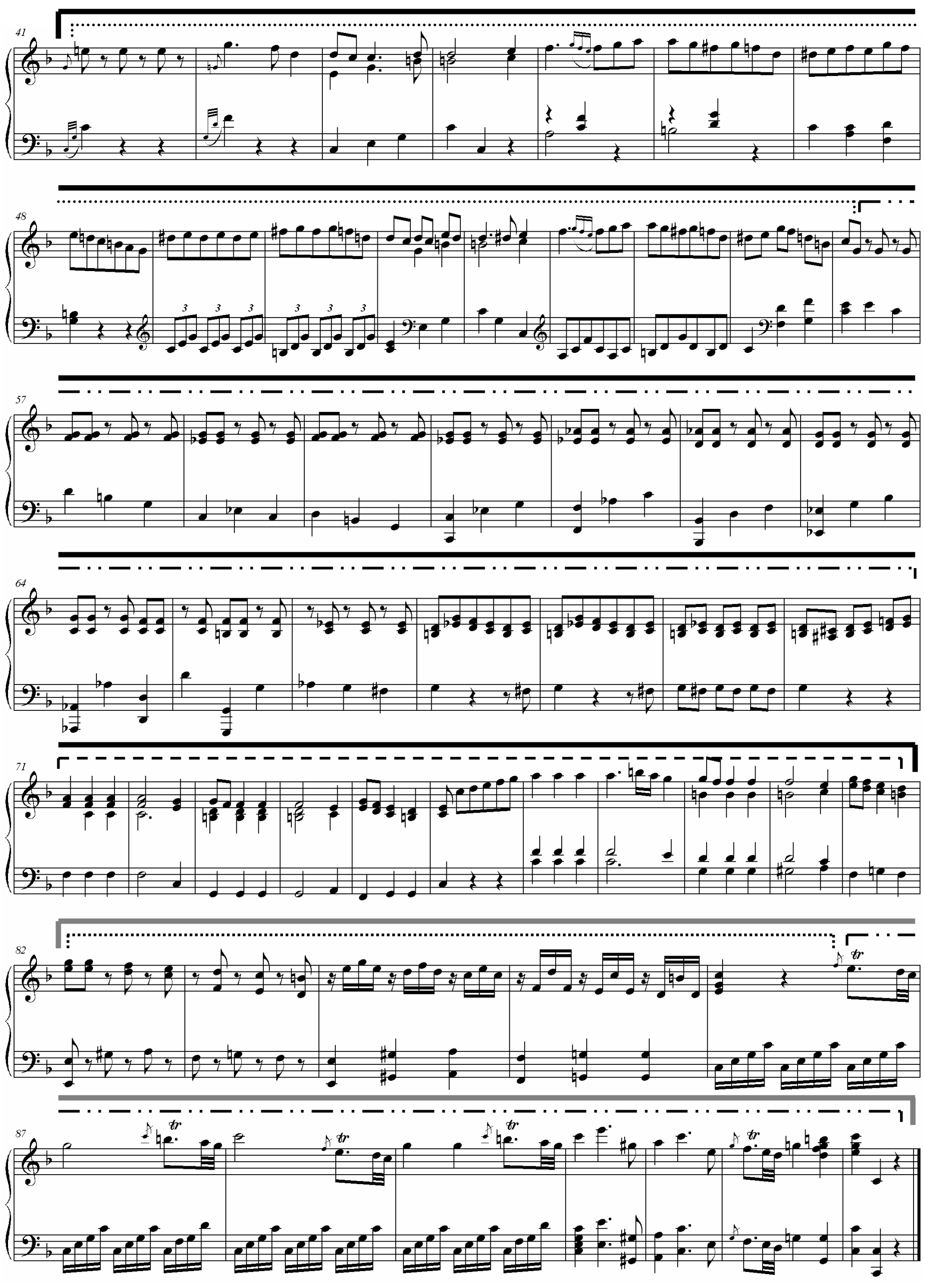
The aforementioned structure is shown through PNs in Fig. 11. The three music objects that come into play are: **Exposition** (meas. 1-93), **Development** (meas. 94-132), and **Recapitulation** (meas. 133-229). This first example originates a very simple Petri Net, which is a trivial sequence of three steps. The description is very compact and clear from a graphical point of view, but it conveys poor musicological information. Besides, intrinsic characteristics of Music PNs (such as parallelism) are not exploited. The example in Fig. 11 will be taken up later, after showing how recapitulation can be derived from exposition, and will originate a more interesting net.



Key:

━━━━	First Group (**FG**)
⋯⋯⋯	main theme, first theme in FG (**1FG**)
— · · —	second theme in FG (**2FG**)

═══	Transition (**T**)
⋯⋯⋯	**1T**
— · · —	**2T**
- - - -	**3T**
— · —	**4T**
═══	**5T**
———	**6T**
— — -	**7T**



Key:

▬	Second Group (**SG**)	▬	Close Group (**CG**)
··········	secondary theme, first theme in SG (**1SG**)	··········	first theme in CG (**1CG**)
— · · —	second theme in SG (**2SG**)	— · · —	second theme in CG (**2CG**)
– – – –	third theme in SG (**3SG**)		

At a lower degree of abstraction, we now provide a closer look at the inner structure of the exposition. In Fig. 12, the exposition follows the upper branch of the net, whereas the lower one represents thematic transformations applied in the recapitulation. As noted before, in sonata exposition there is a first group of themes (**1FG** and **2FG** are the first and the second theme of the group, respectively), then a transitional segment (**T(1)** and **T(2)**), and finally a second group of themes (**SG**), followed by a close group (**CG**). The lower branch shows the evolution of the same thematic material in the context of recapitulation. In this case, our PN illustrates in a clear and compact way the similarities between sections of a sonata. We notice that:

- **1FG** and **2FG** recur in both sections, without variations.
- The transitional material is slightly different, and those dissimilarities will be further investigated.
- Finally, **SG** and **CG** are identical in the exposition and in the recapitulation, but, in the second case, they undergo a transposition. In order to apply this operator to the music object only in the recapitulation, we first have to load it (**Load SG/CG** place) and then to execute it, with or without transposition (**Exec SG/CG** place).

The grouping structures we are using refer to smaller blocks of measures, coming from the analytical process of segmentation proposed in [11]. This process has identified the main theme, the secondary theme and other themes from first, second, and closing groups.

In order to make the reading of Music PNs easier, in Figure 12 and following places with an associated music object are gray colored, whereas places without an associated music object (as either inherit a transformed music object from the preceding places or simply make PN evolve without musical consequences) present a white background color.

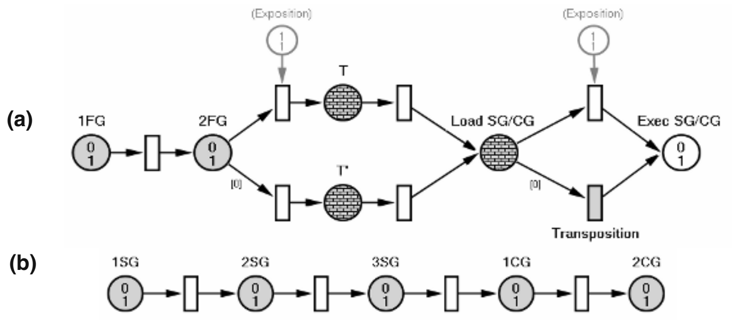


Fig. 12. (a) Themes evolution in Exposition and Recapitulation; (b) SG/CG subnet

Thanks to the aforementioned considerations, now we can revise Fig. 11 in order to synthesize **Recapitulation**'s dependency from **Exposition**. The result is shown in Fig. 13, where high-level structural representation is even more compact.

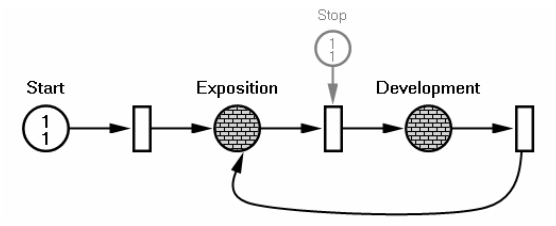


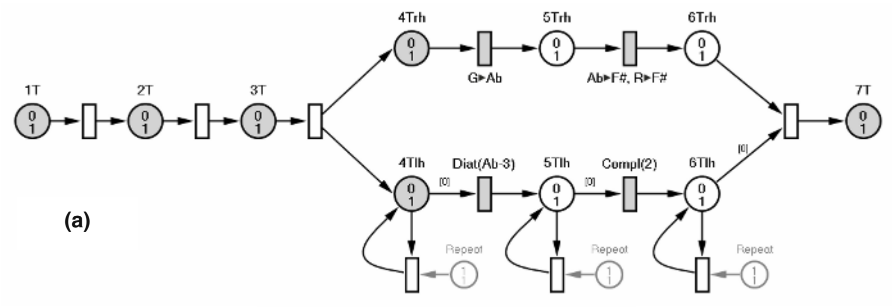
Fig. 13. A revised representation of sonata form

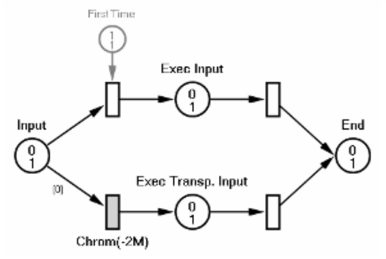
Finally, we take into consideration what happens in the transition between the first and the second group of themes. In the exposition and in the recapitulation a different behavior will occur, as tonal relationships between the first and the second group of themes change (passing from tonic-dominant to tonic-tonic).

The complete example, shown in Fig. 14, illustrates the expressive power of PN formalism. The choice of music objects to consider reflects only one of many alternatives: our goal here was keeping graphic complexity low, showing – nevertheless – a number of relationships among original music materials. Of course, other grouping structures are conceivable, and they would embrace other properties and relationships of music material. For instance, we identified **1T** and **2T** as different objects, because they are treated in a slightly different way in exposition and in recapitulation sections, and not for musicological reasons. Likewise, within **1T** the rhythmic and melodic figures played by the left hand at measure 23 and 24 are very similar, but this relationship was intentionally ignored in order to design a more compact Music PN. At a lower degree of abstraction, it is worth noting that many other relationships can be caught; unfortunately, this would imply a more complex (and less readable) Petri Net.

Fig. 14(a) illustrates music evolution in the first occurrence of transition, i.e. the transition segment within exposition. After 6 measures (**1T** and **2T**), we had to split music score in two parts, denoted by suffix **rh** (standing for right hand) and **lh** (for left hand) respectively. Concerning **3T**, it is considered a single music object within exposition, while it is managed as two separated parts during recapitulation section.

The splitting mechanism previously introduced is very effective when applied to **4T**, **5T** and **6T**: in fact, it lets us describe measures 31-36 as the transformation of the same elementary music material. We underline that – over the two branches – a music object is associated only to **4Trh** and to **4Tlh** (grey places), whereas the subsequent white-colored places inherit music objects from the preceding ones. Of course, transitions operate some changes: concerning right hand, music material undergoes a number of pitch substitutions (e.g., passing from **3Trh** to **4Trh**, G becomes A*b*); for the left part, the minimal grouping structure is made of only one measure, always literally repeated twice. In our models, the transitions that have an associated algorithm are grey-colored. In this case, the musical operators associated to transitions are: diatonic transposition **Diat(A*b*-3)**, two degrees (i.e. a third) down along A*b* major scale, and complement **Compl(-2)** to obtain the second inverted chord. After **6T**, the evolution of Music PNs brings to converge at **7T**.





Subnet for Fig. 14(b)

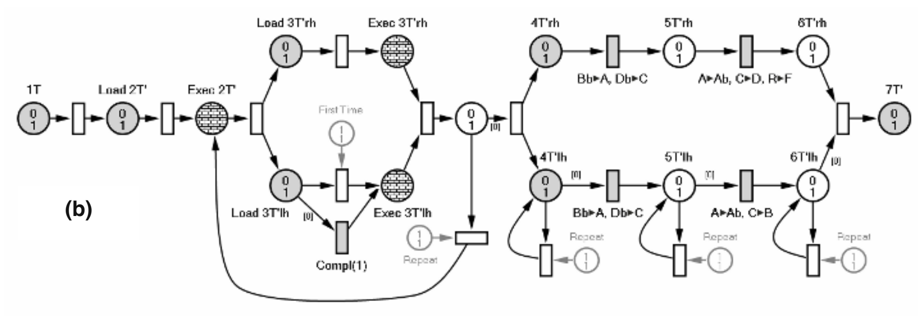


Fig. 14. (a) Transition from Exposition; (b) Transition from Recapitulation

In Figure 14 we provide two distinct models (a and b respectively) to illustrate how transition evolves in exposition and in recapitulation. However, the two parts of Figure 14 are not independent: the original material is the same in both cases, as indicated by the place names. **1T** remains unchanged, whereas many other music objects undergo a transposition from the dominant degree to the tonic degree. The harmonic grid cannot be the same in both cases, and this consideration emerges clearly looking at the different music operators associated to PN transitions in Figure 14(a) and (b). Together with **Compl** and **Diat** operators, already introduced for Figure 14(a), in the subnet related to Figure 14(b) we find a **Chrom(-2M)** operator, that transposes all the corresponding intervals a major second below.

A single PN describing the two occurrences could be created, as we did in Figure 12, but the result would be less readable.

In order to appreciate the operation of PN transitions, the following example (Figure 15) relates a significant part of Figure 14(b) to the corresponding measures in the recapitulation.

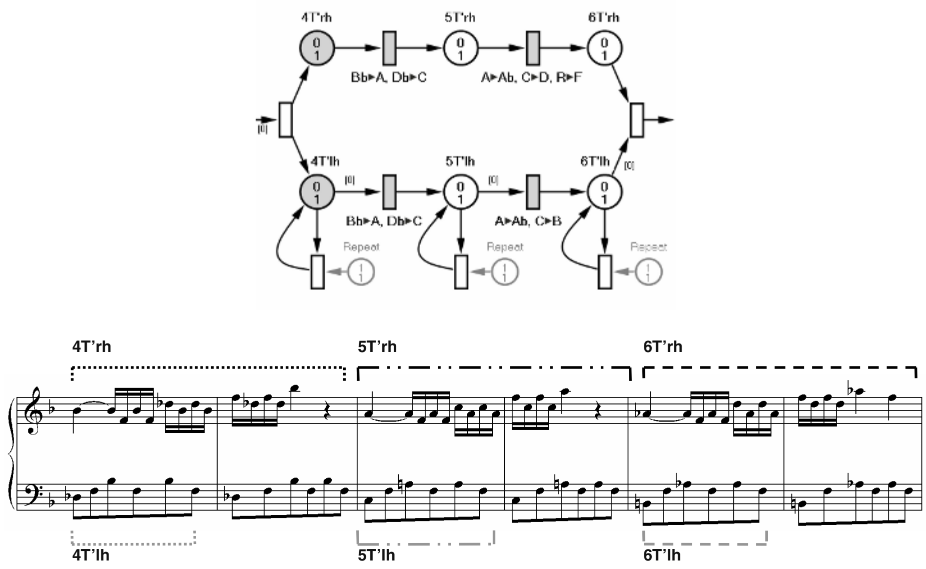


Fig. 15. Operations performed by PN transitions

6 Formats to Collect PN-Oriented Analysis Results

The Petri Net Markup Language (PNML) is a proposal of an XML-based interchange format for Petri Nets. The standardization effort originated the working draft of ISO/IEC 15909-2 for a Transfer Format for High-level Petri Nets. At present, the PNML format is supported by several PNs tools, which facilitates the exchange of PN models. The goal of this format is the possibility to add new extended PNs types, maintaining a set of basic features. This allows, for example, the export of a PN model to another tool that, adopting the PNML format, simply ignores the unsupported features without losing all information.

The PN design tool we have used, namely ScoreSynth (see next section), adopts a PNML-based approach to save and load PN models. In ScoreSynth we have developed an extension to PNML basic PNs types to incorporate all the extensions we have to use for Music PNs.

In order to collect the results coming from a PN-oriented analysis, our purpose is to encode both music symbolic information and the corresponding structural information in a unique data structure. XML provides an effective way to represent musical information at different levels of abstraction. Thanks to the file format we are developing at LIM[2], namely MX, it is possible to represent notational symbols as well as

[2] Laboratorio di Informatica Musicale, Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano.

music objects and related structures. MX is an XML-based format that describes music information according to a multi-layer structure, where each layer is specific to a different degree of abstraction in music information. As described in [5], MX is constituted by General, Structural, Music Logic, Notational, Performance and Audio layers.

Our encoding format is particularly suitable to describe information coming from a manual or automatic segmentation process. Thanks to its multi-layer layout, themes and other musical objects (Structural layer) can easily refer to organized symbols in score (Music Logic layer). Besides, MX format encodes also the relationships intervening among music objects.

Another important characteristic of MX encoding is the possibility to import and/or support fragments from other XML-based formats, such as PNML.

7 Related and Future Works

The application used to design, execute and debug all the PNs involved in this paper is called ScoreSynth, and was developed at LIM. Since 1980, we have been using PNs as the basic tool for music description and processing, creating some applications to support our approach. In our recent work, we have switched to Windows operating system (Macintosh was the preceding platform), and we have developed two new applications: ScoreSynth and MediaSynth. Both are based on the same interface, which allows drawing, editing, and executing PNs. In ScoreSynth, the material associated to places is symbolic and represented in MX format, while MediaSynth is dedicated to multimedia processing, with media associated to places and effects associated to transitions. The PN-based analysis presented in this paper was performed and validated by ScoreSynth. Thus, in our implementation of a tool to design and executing Music PNs, the format chosen to represent music objects is MX, an XML-based encoding that can incorporate all levels of music representation.

Concerning future works, an interesting idea is exploiting analysis results to write new music compositions. For instance, it is possible to maintain the same structural characteristics of the analyzed piece, but changing music objects associated to places and/or music operators associated to transitions. These possibilities will be further explored in the next future.

8 Conclusions

In our opinion, after some opportune extensions, Petri Nets can effectively represent the results of a musicological analysis process. This representation can be more or less effective, readable, and compact depending on the music objects we choose and on the degree of abstraction we want to reach. Music PNs work very well when applied to general structures (e.g. sonata form), provide interesting and peculiar results when applied to thematic segmentation (e.g. themes in a section), and finally present a too complex layout when atomic music objects are involved.

Their full potentialities are exploited when a number of relationships occur in a music piece. PNs are very effective and efficient to describe concurrent, distributed, parallel processes, and an *ad hoc* choice of music objects can highlight such peculiarities. The application of this formal tool to music analysis can show surprising results.

Acknowledgements

The authors wish to acknowledge the partial support of this project by Italian MIUR (FIRB "Web-Minds" project N. RBNE01WEJT_005) and the Italian National Research Council, in the framework of the research program "Methodologies, techniques, and computer tools for the preservation, the structural organization, and the intelligent query of musical audio archives stored on heterogeneous magnetic media", Finalized Project "Cultural Heritage", Subproject 3, Topic 3.2, Subtopic 3.2.2, Target 3.2.1. We also want to acknowledge the members of the IEEE MX WG (PAR1599) for their cooperation and interest in our work. This work has been made possible by the efforts of researchers and graduate students of LIM.

References

1. De Matteis, A., Haus, G.: Formalization of Generative Structures within Stravinsky's The Rite of Spring. In: Journal of New Music Research, Vol. 25, N. 1. Swets & Zeitlinger B.V., Amsterdam (1996) 47–76
2. Haus, G., Rodriguez, A.: Music Description and Processing by Petri Nets. In: 1988 Advances on Petri Nets, Lecture Notes in Computer Science, N. 340. Springer-Verlag, Berlin (1989) 175–199
3. Haus, G., Rodriguez, A.: Formal Music Representation; a Case Study: the Model of Ravel's Bolero by Petri Nets. In: Haus, G. (ed.): Music Processing. Computer Music and Digital Audio Series. A-R Editions, Madison (1993) 165–232
4. Haus, G., Sametti, A.: Modeling and Generating Musical Scores by Petri Nets. In: Languages of Design, Vol. 2, N. 1. Elsevier Publ., Amsterdam (1994) 7–24
5. Ludovico, L.A., Haus, G.: Music Segmentation: an XML-Oriented Approach. In: CMMR 2004 Post-symposium Proceedings, Lecture Notes in Computer Science. Springer-Verlag, Berlin (2004)
6. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. The MIT Press, Cambridge (1983)
7. The New Grove Concise Dictionary of Music. Macmillan Publishers, Basingstoke (1994)
8. Mozart, W.A.: Sonata in F major KV332 (300k). In: Complete Piano Sonatas. G. Henle Verlag, München (2005)
9. Petri, C.A.: General Net Theory. In: Proceedings of the Joint IBM & Newcastle upon Tyne Seminar on Computer Systems Design (1976)
10. Peterson, J.L.: Petri Net Theory and the Modelling of Systems. Prentice Hall, New Jersey (1981)
11. Surian, E.: Manuale di Storia della Musica, Vol. 2. Rugginenti Editore, Milano (1992) 511–513

A Review on Techniques for the Extraction of Transients in Musical Signals

Laurent Daudet

Laboratoire d'Acoustique Musicale, Université Pierre et Marie Curie (Paris 6),
11 rue de Loumel, 75015 Paris, France
daudet@lam.jussieu.fr
http://www.lam.jussieu.fr

Abstract. This paper presents some techniques for the extraction of transient components from a musical signal. The absence of a unique definition of what a "transient" means for signals that are by essence non-stationary implies that a lot of methods can be used and sometimes lead to significantly different results. We have classified some amongst the most common methods according to the nature of their outputs. Preliminary comparative results suggest that, for sharp percussive transients, the results are roughly independent of the chosen method, but that for slower rising attacks - *e.g.* for bowed string or wind instruments - the choice of method is critical.

1 Introduction

A large number of recent signal processing techniques require a separate processing on two constitutive components of the signals : its "transients" and its "steady-state". This is particularly true for audio signals, by which we mean primarily music but also speech and some environmental sounds. Amongst all applications, let us mention : adaptive audio effects (enhancement of attacks [1], time-stretching[2], ...), parametric audio coding [3] (the transients and the steady-state are encoded separately), audio information retrieval (transients contains most of the rythmic information, as well as specific properties for timbre identification). Furthermore, transients are known to play an important role in the perception of music, and there is a need to define perceptually-relevant analysis parameters that characterize the transients.

However, there is a plethora of methods for the Transient / Steady-State (TSS) separation, with little indication of their relative merits. This arises from the fact that there is no clear and unambiguous definition of what a "transient" is, not what "steady-state" means for musical signals that are by essence non-stationary. In mathematical terms, this is an ill-posed problem, that can only lead to some tradeoffs. Indeed, we shall see that every definition leads to a specific decomposition scheme, and therefore different results in the separated TSS components. The goal of this paper is to make a review of some commonly used techniques, together with comparative results. Some of these methods are recent developments, but others, that are well described in the literature, are mentioned here for the sake of completeness. Although we do not claim any sort of exhaustivity, we hope that we have covered the most important ones that have been successfully used for music. Obviously, a lot of other techniques could be used, that are not described in this article, since they were developed for other classes of signals (*e.g.* transient detection in duct flows, in underwater acoustics, in engine sounds). Our choice is to focus on musical signals, with a special emphasis on methods where the author had some hands-on experience, and where the computational complexity is reasonable so that they could be realistically applied to real full-band audio signals.

At this point, we should make it clear that the problem of TSS separation is related to, but distinct from, other classical musical signal processing tasks that are the classification of segments into transients or steady-state (for instance the way it is done in subband audio codecs such as MPEG 1 layer III, for the decision between the long and short window mode), or the binary TSS segmentation in time. On the contrary, all methods suggested here assume an *additive* model for the sounds, where transients and steady-state can in general exist simultaneously. TSS separation is also different from the problem of note onset detection [4], although it is clear that they share common methods.

The different methods can be grouped into 3 classes (even though this classification is certainly not unique), depending on the structure of their outputs (see table 1).

The fist class of methods, amongst the simplest in their principle, are based on linear prediction (section 2). They provide a decomposition of the sound into its excitation signal and a resonating filter. If the filter has been well estimated, most of the energy of the excitation signal is located at attack transients of signals.

The second class of methods (section 3) do not define transients directly, but rather extract from the signal its "tonal" part (also called sinusoidal part). If this extraction is successfully applied, the residual signal exhibits, as in the linear predictions methods above, large bursts of energy at attack transients. It also contains some slowly-varying stochastic residual.

Finally, the last class of signals (section 4) assume some explicit model for the transients, and the output of the model is 3 signals, that can be summed to reconstruct the original : one for the Sinusoidal part, one for the transients, one for the Residual (these models are often called STN models, for Sines + Transients + Noise).

Table 1. Different transient extraction methods can be classified according to their outputs. For each class of methods, the signal related to the transients is highlighted.

Method	Outputs			Section
Linear prediction	Resonance filter coefficients		**Excitation signal**	2
Tonal extraction	Tonal signal	**Non-tonal signal**		3
STN models	Tonal signal	**Transients signal**	Noise signal	4

Some results are presented in section 5, where some of the above approaches are compared on test signals. A tentative guide for the choice of a method most suitable for the problem at hand is finally presented, based on their pros and cons, that have to be balanced with computational complexity. The last section of this article (section 6) presents conclusions and future directions for research.

2 Methods Based on Linear Prediction

In this class of methods, the distinction between transient and steady-state is related to the notion of *predictability*. A steady-state portion of the signal is one where any part of this segment can be accurately predicted as soon as some small sub-sequence (the training sequence) is known.

Linear prediction in the time domain is a widely-used technique, since it is typically the core of most speech coding technologies. An the simplest auto-regressive (AR) case, the underlying idea is to consider the sound as the result of the convolution between an excitation signal and an all-pole filter. The Yule-Walker equations allow the estimation of the best order-P filter, that minimizes the energy of the prediction error. Once the filter is estimated, the excitation signal is simply the result of the filtering of the signal by the inverse filter (which only has zeros and therefore is stable). For steady-state parts of the sounds, the excitation signal can usually be simply modeled, for instance as an impulse train for voiced phonemes. More generally, this excitation signal will have a small local energy when the signal is highly predictable (steady-state portions), but energetic peaks when the audio signal is poorly modeled by the AR model. This typically corresponds to non-predictible situation, such as attack transients (or decay transients *e.g.* in case of a damper).

The obtained decompositions has a physical interpretation for source - filter models : when the excitation has a flat frequency response (impulse or whte noise), the AR filter is a good estimate of the instrument's filter. In the more general case when the excitation does not have a flat spectrum, it nevertheless provides a qualitative description of temporal and spectral properties of the sound, even though the physical interpretation is strictly lost.

Usually, this method gives good results when the signals are the result of some (non-stationary) excitation, filtered and amplified by a resonator. However, it has strong limitations : first, the order estimation (order of the filter) has to be known or estimated beforehand, which can be a hard task. Second, the estimation of the resonant filter will only be successful only if the training sequence does not contain a large portion of transients, and this makes the estimation on successive notes sometimes not reliable. However, this method is very well documented and easy to use in high-level DSP environments such as Matlab, and can be quite accurate on isolated notes.

Further extensions can be designed, for instance with auto-regressive with moving average (ARMA) models, but in this case the complexity is increased, both for the parameter estimation and for the inverse filtering that recovers the excitation signal.

3 Methods Based on the Extraction of the Tonal Content

In this class of methods, as in the linear prediction methods above, there is no explicit model for the transients. The goal here is to remove from the signal its so-called “tonal” or “sinusoidal” components. The residual is then assumed to contain mostly transients.

3.1 Segmentation of the Short-Time Fourier Transform

A natural way of expanding the above extraction methods is by using time-frequency analysis. The simplest implementation is the Short Time Fourier Transform (STFT), which provides a regularly-spaced local frequency analysis. Now, within each frequency subband, it is possible to perform the prediction search within each frequency band. The simplest model is based on the so-called “phase vocoder” [5], originally designed to encode speech signals. For the task of TSS separation, each time-frequency discrete bin will be labelled as ”transient” (T) or ”steady-state” (SS), and the underlying assumption is that we can neglect the influence of time-frequency bins that have a significant contribution in both domains. Note that we keep the terminology “steady-state” employed in the original papers, although the word “tonal” would be here more appropriate. After labeling, each signal, transient or steady-state, is reconstructed using only the corresponding time-frequency bins.

The simplest criteria for the identification of tonal bins is based on phase prediction in a given frequency bin k [5]. On steady-state portions of the sound, the (unwrapped) phase ϕ_n (n stands for the index of the time window) evolves linearly over time (hence the definition of instantaneous frequency as time derivative of the phase). Now, one looks at predicting the value of the phase ϕ_n in the current window, knowing its past values. A first-order predictor gives :

$$\phi_n^{pred} = 2\phi_{n-1} - \phi_{n-2} \tag{1}$$

Now, ϕ_n^{pred} is compared to the measured ϕ_n, and the labeling is based on the discrepancy between these two values :

$$\text{Time-frequency bin } (k,n) \text{ of type } \begin{cases} \text{SS} & \text{if } |\phi_n^{pred} - \phi_n| < \varepsilon \\ \text{T} & \text{otherwise} \end{cases} \tag{2}$$

where ε is a small constant that defines the tolerance in prediction error, for instance due to slight frequency changes, and it has to be adapted to the analysis hop size (number of samples between two analysis windows). Obviously, this method can be seen as an extension of the linear prediction methods (section 2), as here a -basic- predictor is applied in every frequency bin.

More recently, this method has been refined in a few directions. It has been shown [2] that the results are significantly improved when processing the results in different subbands (with increasing time resolution at high frequencies), as well as by using an adaptive threshold for ε in equation 2. For onset detection purposes, the method has been further extended by using a complex-valued difference [6] that not only takes the phase difference into account, but also the magnitude (attack onsets are usually characterized by large jumps in amplitude). Finally, a magnitude-based modeling in [7] tries to fit a triangular shape at transients for a given STFT frequency bin.

3.2 Methods Based on Parametric Representations: Sinusoidal Models and Refinements

Parametric representations assume a model for the signal, and the goal of the decompositions is to find the set of parameters that allow, at least approximately, to resynthesize the signal according to the model. For speech / music signals, it is natural to assume that the signals are mostly composed of tonal -*i.e.* sinusoidal-components, and here transients are defined as part of the non-tonal residual.

The simplest of this model was originally proposed by McAulay-Quatieri [8] for speech signals, where the sound is seen as a linear combination of a (relatively small) number J of sinusoids :

$$x(t) \approx \sum_{j=1}^{J} A_j(t) \sin(\varphi_j(t)) \tag{3}$$

where $\varphi_j(t) = \int_0^t \omega_j(\tau)d\tau + \varphi_j(0)$ represents the phase of the j-th partial. The parameters $(A_j, \omega_j, \varphi_j)$ for each partial sinusoid are assumed to evolve slowly over time, hence their values only have to be estimated frame-by-frame.

When applied to music signals, the residual contains all the components that do not fit into the model : stochastic components, or fast-varying transients. For general music processing purposes, this model has been refined to take into account the stochastic nature of the residual (Spectral Modeling Synthesis or SMS [9]). More recently, the analysis of this residual has been embedded in a statistical framework [10] that allows transients detection and modelling.

3.3 Methods Based on Subspace Projection

High resolution methods allow a precise estimation of exponentially damped sinusoidal components in complex signals. As in the linear prediction case, this model is physically motivated since exponentially-damped vibrations are the natural free response of oscillatory systems. The model is as follows :

$$x(t) = \sum_{j=1}^{K} A_j z_j^t + n(t) \tag{4}$$

where A_j is a complex amplitude, $z_j = e^{\delta_j + i2\pi f_j}$ is a complex pole that represents both the oscillation at frequency f_j but also the damping through the δ_j term, and $n(t)$ is a noise term that is assumed white and gaussian. The principle of high resolution techniques is to estimate the values of all z_j through numerical optimization techniques. The obtained resolution is typically much higher than in the simple Fourier case. Specific methods have been proposed, that offer a better robustness to noise than the standard Pisarenko or Prony estimations methods. In particular, ESPRIT [11] and MUSIC [12] reduce this task to an eigenvector problem. When the number K of components is known, the span of the K obtained eigenvectors is called the “signal space”, its complement is called the “noise space”. Projecting the signal onto these 2 subspaces provides a natural decomposition of the signal into its tonal and non-tonal components. YAST [13], a recent variant of these methods for time-varying systems, is particularly suitable for music signals since it allows a fast processing of large-size signals. Note that the white noise hypothesis is generally not verified, in which case the processing has to be performed in separate subbands. Also, the number of components has to be known or estimated, and therefore the best estimates are obtained on isolated notes, where the number of components does not change over time.

4 Sines + Transients + Noise Models

Three-components decompositions of the sounds have become very popular in audio coding, especially in the framework of MPEG-4. The aim is to decompose the file into three additive components, the tonal or sinusoidal component, the transients, and a slowly-varying wide-band stochastic component called “noise”. The extraction can be done sequentially, first by a tonal component extraction as in section 3, and then by a transient processing on the non-tonal part. Alternatively, the separation can be done simultaneously, which usually gives better results but requires more computational power.

4.1 Sequential Estimation of Tones with Hybrid Dictionaries

In this TNS framework, the simplest method for TSS is to estimate each component at a time, first transient and then steady-state, or vice-versa. This is the basis for hybrid methods, that make use of two different orthogonal transforms. In the Transient Modeling Synthesis (TMS) scheme [3], the tonal part is first extracted by taking the large coefficients of a Modified Discrete Cosine Transform (MDCT). In a dual way, transients are analyzed in a pseudo-time domain constructed by taking the Fourier transform of the discrete cosine coefficients.

In [14], the tonal part is first estimated using the largest Modified Discrete Cosine Transform (MDCT) coefficients of the signal. Transients are then estimated by the largest Discrete Wavelet Transform (DWT) coefficients of the signal.

These methods are very simple, but suffer from two drawbacks: first, each component (T or SS) biases the estimate of the other component ; and second, at each stage the choice of the threshold between large “significant” coefficients, and small “residual” coefficients is difficult. Although they may provide satisfactory results in some simple cases, the above limitations call for a simultaneous estimation of both TSS components, which is the topic of the next sections.

4.2 Simultaneous Estimation by Adaptive Time-Frequency Resolution

With adaptive time-frequency analysis, it is possible to obtain estimation of different components. the idea is to adapt *locally* the resolution of the transform to the signal. The choice of the resolution is based on some sparsity measure, for instance Shannon-like entropy measures. A simple version of this is the Best orthogonal Basis algorithm [15], where a multiresolution transform that has a tree-like structures (for instance, wavelet packets) adapts locally its resolution in time or frequency.

More recently, this idea has been extended and applied to music TSS separation, through "Time-frequency jigsaw puzzles" [16]. Here, this adaption step is made even more locally, in so-called *super-tiles* of the time-frequency plane. The algorithm is run iteratively until some convergence is reached. At every iteration, the algorithm finds, in each super-tile, the optimal resolution, transforms the signal accordingly, and subtracts the largest coefficients. As in many methods above, the choice of the threshold that governs, for a set of transform coefficients at a given resolution, what is "significant" and what is not is a critical point based mostly on empirical evidence.

4.3 Simultaneous Estimation by Sparse Overcomplete Methods

The goal of sparse overcomplete methods is to decopose the signal x as a linear combination of fixed elementary waves, called "atoms" :

$$x = \sum_k \alpha_k \varphi_k \ , \tag{5}$$

where α_k are scalars, and φ_k are the atoms drawn from a dictionary $\mathcal{D}$. In finite dimension, the dictionary $\mathcal{D}$ is said overcomplete when it spans the entire space and has more elements than the dimension N of the space. In this case, there is an infinity of decompositions of the form (5), and one would like to find one that is sufficiently sparse, in the sense that a small number $K \ll N$ of atoms provide a good approximation of the signal :

$$x \approx \sum_{j=1}^{K} \alpha_{k_j} \varphi_{k_j} \ , \tag{6}$$

If the dictionary is composed of two classes of atoms $\mathcal{D} = \mathcal{S} \cup \mathcal{T}$, where $\mathcal{S} = \{g_i\}$ is used to represent the tonal components of the sound (for instance long-window Gabor or Modified Discrete Cosine Transform atoms), and $\mathcal{T} = \{w_i\}$ is used to represent the transient part of the sound (for instance short-windows Gabor atoms, or wavelet atoms), a sparse approximation of the signal will provide a natural separation between transients and tones. In this case, the noise is simply the approximation error, due to components that do not belong to either class, and the tonal layer (resp. the transient layer) is the partial reconstruction in the signal using only atoms in $\mathcal{S}$ (resp. atoms in $\mathcal{T}$).

However, for general overcomplete dictionaries, finding a good sparse approximation is a non-trivial task, and indeed it has been shown that finding the optimal K-terms approximation of the signal x is a NP-hard problem [17]. Many recent signal processing techniques have emerged recently (Basis Pursuit, Matching Pursuit, FOCUSS, ...), and we will here only a few of them that have been specifically applied to the TSS problem.

Matching Pursuit and Extensions. The Matching Pursuit [18] is an iterative method that selects one atom at a time. At every iteration, it selects the "best" atom φ_{k_0}, *i.e.* the one that is the most strongly correlated with the signal: $k_0 = \arg\max_k |\langle x, \varphi_k \rangle|$. The corresponding weighted atom is then subtracted from the signal $x \leftarrow x - \langle x, \varphi_{k_0} \rangle \varphi_{k_0}$ and the algorithm is iterated until some stopping criteria is reached (*e.g.* on the energy of the residual). This algorithm is suboptimal in the sense that, although it chooses at every iteration the atom that minimizes the residual energy, there is in general no guarantee that the set of selected atoms provide the best sparse approximation of the form (6). However, Matching Pursuit has become quite popular, mainly due to its simplicity, but also because experimental practice shows that in most cases the obtained decompositions are close to optimal (at least for the first few iterates).

For the sake of TSS separation, this algorithm has been extended into the Molecular Matching Pursuit [19], that at every iteration selects a whole group of neighboring atoms, called "molecule". The dictionary $\mathcal{D}$ is, as in the above hybrid model, the union of a MDCT basis (for tones) and a DWT basis (for transients), and a molecule is only composed of one type of atoms. Besides a reduced computational complexity, selecting molecules improves significantly the TSS separation over the original matching pursuit: first, it prevents isolated large atoms to be tagged as significant ; and second, it forces low-frequency large-scale components, that by themselves could equally go into transients or tones, into only one of these components according to the local context.

Global Optimization Techniques. When the above results are not satisfactory, it may be desirable to use algorithms that choose a globally optimal or near-optimal solution, for a given optimality criteria. The problem can usually be written as a minimization problem:

$$u = \arg\min_u ||x - \Phi u||_2^2 + \lambda ||u||_p^p \qquad (7)$$

where Φ is the (rectangular) matrix of our overcomplete basis, and λ is a scaling parameter for the sparsity measure $||u||_p^p = \sum_{k=1}^{M} |u_k|^p$, with $0 < p \leq 1$.

The resolution of this problem typically involves very high computational costs. Many such techniques have been proposed in the literature, such as Basis Pursuit or FOCUSS. Amongst them, the Fast Iterated Reweighted SParsifier (FIRSP) [20] has been successfully applied to audio TSS separation. The principle is to use the reweighted least squares algorithm for the optimization problem (7) with $p \to 0$ (this enforces a strong sparsity). For a practical implementation, this problem is reset in the Expectation Maximization (EM) framework.

In general, this requires at every iteration the inversion of the matrix Φ, which can be very costly for large matrices.

However, this algorithm is of real practical use when the dictionary is the union of orthonormal bases, since in this case each EM iteration is replaced by a series of Expectation Conditional Maximizations within each orthonormal subspace, and the above matrix inversion reduces to a scalar shrinkage. Here, the strongly reduced computational complexity makes it a realistic choice for processing long audio segments. Further details can be found in [20, 21]. For TSS separation, the union of 4 orthonormal bases has been used, a long-window Modified Discrete Cosine Transform (MDCT), a long-window Modified Discrete Sine Transform (MDST), a short-window MDCT and a short-window MDST. After a few FIRSP iterations, the transients' energy is mostly concentrated on the short-windows transforms, and the SS in the long-windows transforms. Note that other orthogonal transforms can be used, for instance discrete wavelets for the transients ; however, preliminary results suggest that results are quite similar on most typical test signals.

5 Comparative Results

We have compared extraction results for 3 recent methods

- the YAST high-resolution method (paragraph 3.3). The signal is processed in 2500 Hz equal-width subbands, with 10 to 20 sinusoidal components par subband;
- the adaptive phase-vocoder approach (paragraph 3.1), described in [2];
- the jiigsaw puzzle approach (paragraph 4.2), in the variant TFJP2 described in [16].

Two soundfiles were tested, that have very different transient behavior: a glockenspiel, where the attacks are very sharp (the energy rising time is of a few ms); and a trumpet, where the energy rises on much longer time-scales (typically 50 ms, but this can extend to much higher values).

Separation results are presented in figures 1 and 2. On the glockenspiel example (figure 1), the results are quite similar for all three methods : the transient components exhibits sharp, high-amplitude peaks at note onsets. This is the typical case where all the definitions roughly agree on what a transient is.

On the contrary, the trumpet example exhibits very significant differences between the two methods. The YAST algorithm provides large energy bursts at the onset of notes, and smaller ones at their termination. The adaptive phase vocoder tends to capture more of the subtle variations within a note (see for instance the vibrato in the 6th note starting at about 1.5 s). Results of the jigsaw puzzle method are more difficult to interpret: even though its energy is well located on each onset, the amplitude of each onset transient varies somehow unexpectedly (see for instance the difference between the two notes starting at

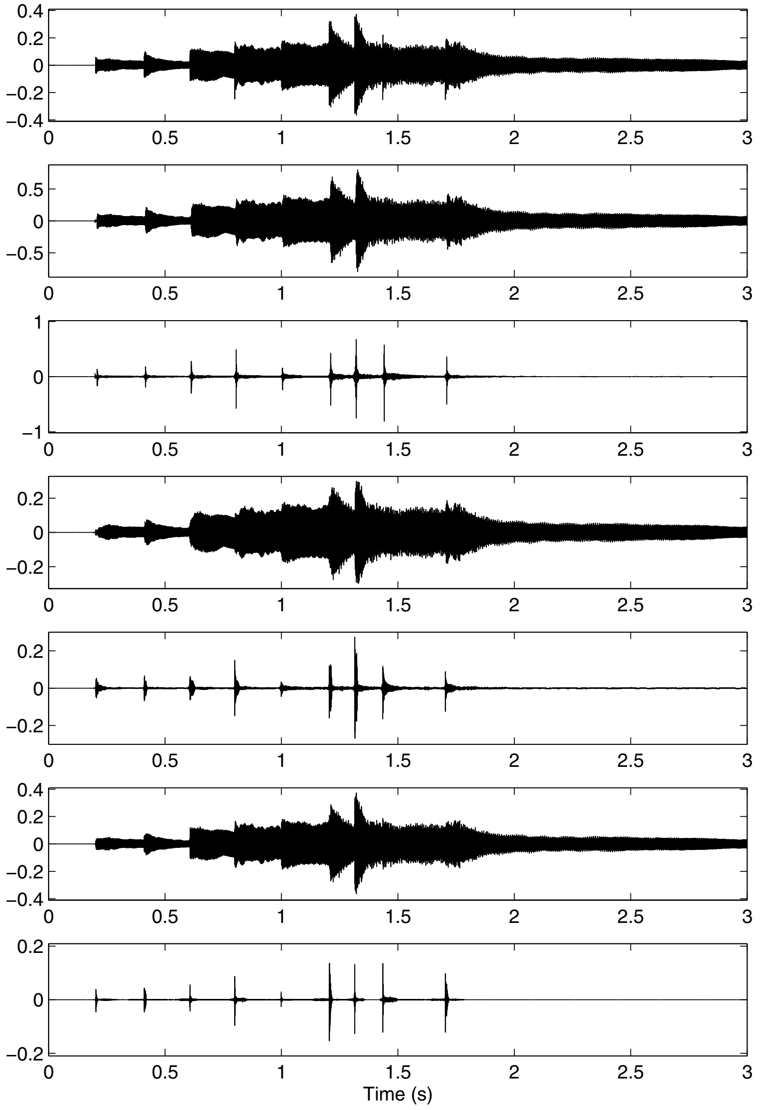


Fig. 1. Glockenspiel signal: comparison of three transients extraction techniques. From top to bottom:original signal, tonal part obtained by YAST, transients obtained by YAST, tonal part obtained by the adaptive phase-vocoder, transients obtained by the adaptive phase-vocoder, tonal part obtained by the jigsaw puzzles, transients obtained by the jigsaw puzzles.

about 1s and 1.3 s). This lack of shift invariance is probably due to a particular choice of super-tiles.

It should be emphasized that these results are only indicative of the relative merits of each method. Results are highly signal-sensitive, at there is no

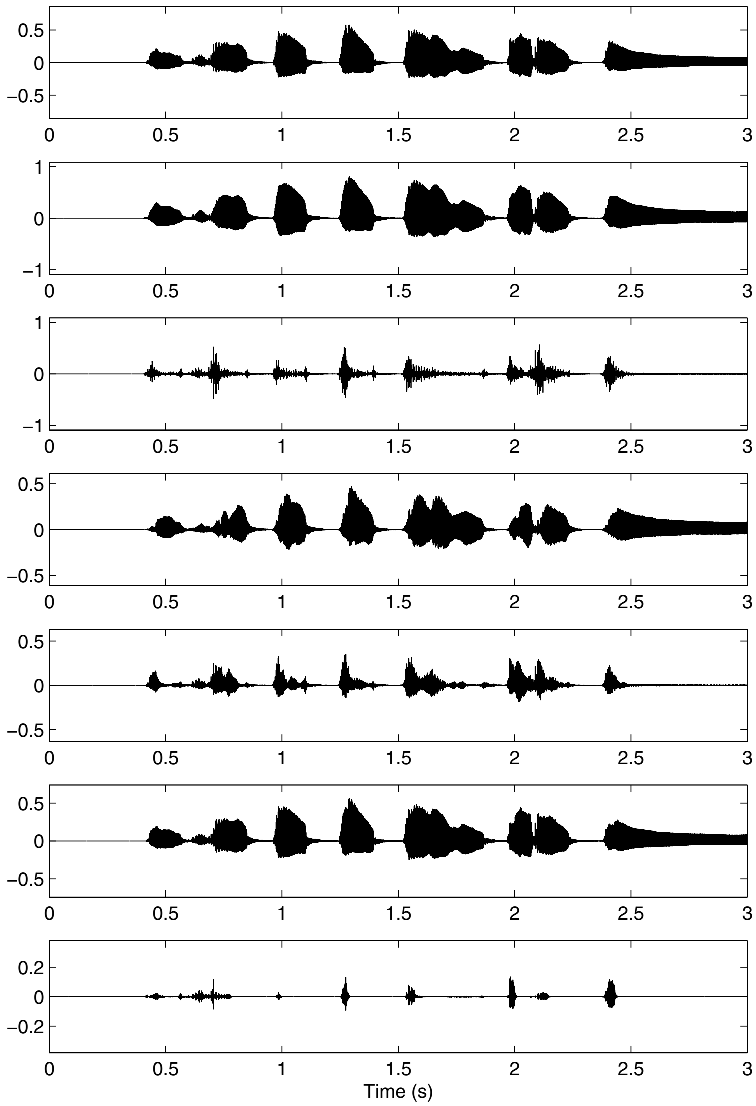


Fig. 2. Trumpet signal: comparison of three transients extraction techniques. From top to bottom: original signal, tonal part obtained by YAST, transients obtained by YAST, tonal part obtained by the adaptive phase-vocoder, transients obtained by the adaptive phase-vocoder, tonal part obtained by the jigsaw puzzles, transients obtained by the jigsaw puzzles.

guarantee that the performance of one algorithm will be similar on two signals that apparently belong to the same class. Furthermore, most of these methods requires a precise fine-tuning of the parameters, and for some of them the results are very sensitive to a particular choice of parameters.

Table 2. Tentative classification of relative computational complexity (from −−: very complex to ++: very fast), pros (P) and cons (C) for each method, as well as their most natural field of application

Method	**Complexity**	**Pros, Cons & applications**
Linear prediction	+	P Source-filter interpretation C Relevant only for flat spectrum sources → *Physical models, lossless coding*
Model for sines only: transients in residual		
Adaptive phase-vocoder	+	P musically relevant C redundancy → *Audio effects, Preprocessing*
Sinusoidal model / SMS	+−	P Explicit signal model C No model for the residual → *Parametric coding, Audio effects*
Subspace methods	−	P High precision C Often requires hand-tuning → *Signal analysis, Preprocessing*
Model for sines and transients		
STN sequential estimation in orthonormal bases	++	P Fast algorithms C Difficult interpretation Threshold choices → *Transform coding*
STN simultaneous estimation by adapted fime-frequency tiles	+−	P General method C Threshold choices, no shift-invariance → *Analysis, Source separation ?*
STN simultaneous estimation by Matching Pursuit / Molecular MP	−	P Generates sparse data (and structured for MMP) C Optimality not guaranteed → *Parametric coding Source separation*
STN simultaneous estimation by global optimization (BP, FIRSP, ...)	− to −−	P Very general, optimality criteria C Potentially very slow → *Analysis, Transform coding ?*

However, we have tried in table 2 to summarize the main advantages and drawbacks of every method presented above. It should be emphasized that this comparison is only relevant within a category : linear prediction methods (described in section 2), Tonal extraction (described in section 3) or STN models (described in section 4). For each category, the balance between computational complexity and relevance of the results should help us in the choice of the most appropriate method for the problem at hand.

6 Conclusion

This paper is a first attempt to review and classify some techniques for the estimation of transients in music signals. Preliminary tests have been conducted; although a systematic comparative test is yet be performed, with more methods and more sound examples. However, a recurrent difficulty for such comparison is the lack of a common platform for testing : one of our medium-term goals is to develop such a software that could act as a unique front-end for some of the numerous methods above. Eventually, the main problem for the task of comparing many TSS techniques is that one has to define one (or more) optimality criteria for deciding when a method is better than another. Some transientness criteria such as described in [1, 22] may be a first step towards relevant efficiency criteria.

One of our findings is that, unsurprisingly, the problem of TSS separation is indeed very different according to the nature of the signal. For sharp percussive sounds, the separation results are roughly independent of the chosen method -the simpler the better !-, but for slower rising attacks - *e.g.* for bowed string or wind instruments - the choice of method is critical. Finally, the biggest challenge is probably to link all these techniques to some perceptually relevant features, since numerous studies on music perception and timbre identification confirm the utmost importance of fast-varying transients.

In the future, there is a need to develop a deeper understanding of the different time-scales involved in human perception. Finding perceptually-relevant signal parameters for transients is in our opinion one of the forthcoming challenges in the musical signal processing field.

Acknowledgements

The author wishes to thank Emmanuel Ravelli, Roland Badeau and Florent Jaillet for their help in the numerical examples. This work was supported by the French Ministry of Research and Technology, under contract ACI “Jeunes Chercheuses et Jeunes Chercheurs” number JC 9034.

References

1. Goodwin, M., Avendano, C.: Enhancment of audio signals using transient detection and modification. In: Proc. AES 117th Conv., San Francisco, CA (2004)
2. Duxbury, C., Davies, M., Sandler, M.: Separation of transient information in musical audio using multiresolution analysis techniques. In: Proc. Digital Audio Effects (DAFx'01), Limerick, Ireland (2001)
3. Verma, T., Levine, S., Meng, T.: Transient modeling synthesis: a flexible analysis/synthesis tool for transient signals. In: Proc. of the International Computer Music Conference, Greece (1997)
4. Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.: A tutorial on onset detection in music signals. IEEE Transactions on Speech and Audio Processing (to appear)

5. Zölzer, U., ed.: DAFX - Digital Audio Effects. John Wiley and Sons (2002)
6. Bello, J., Duxbury, C., Davies, M., Sandle, r.M.: On the use of phase and energy for musical onset detection in the complex domain. IEEE Signal Processing Letters **11** (2004)
7. Rodet, X., Jaillet, F.: Detection and modeling of fast attack transients. In: Proceedings of the International Computer Music Conference, Havana (2001)
8. McAulay, R., Quatieri, T.: Speech analysis/synthesis based on a sinusoidal representation. IEEE Trans. on Acoust., Speech and Signal Proc. **34** (1986) 744–754
9. Serra, X., Smith, J.O.: Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. Computer Music Journal **14** (1990) 12–24
10. Thornburg, H., Gouyon, F.: A flexible analysis-synthesis method for transients. In: Proceedings of the International Computer Music Conference, Berlin (2000)
11. Roy, R., Kailath, T.: ESPRIT - estimation of signal parameters via rotational invariance techniques. IEEE Transactions on Acoustics, Speech and Signal Processing **37** (1989) 984–995
12. Moon, T., Stirling, W.: Mathematical Methods and Algorithms for Signal Processing. Prentice-Hall (2000)
13. Badeau, R., David, B., Richard, G.: Yet Another Subspace Tracker. In: Proc. International Conf. on Acoustics, Speech, and Signal Processing. (2005) 329–332
14. Daudet, L., Torrésani, B.: Hybrid representations for audiophonic signal encoding. Signal Processing **82** (2002) 1595–1617 Special issue on Image and Video Coding Beyond Standards.
15. Coifman, R., Wickerhauser, M.: Entropy–based algorithms for best basis selection. IEEE Trans. Information Theory **38** (1992) 1241–1243
16. Jaillet, F., Torrésani, B.: Time-frequency jigsaw puzzle: Adaptive multiwindow and multilayered gabor expansions. IEEE Transactions on Signal Processing (submitted)
17. Davis, G.: Adaptive Nonlinear Approximations. PhD thesis, New York University (1994)
18. Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing **41** (1993) 3397–3415
19. Daudet, L.: Sparse and structured decompositions of signals with the molecular matching pursuit. IEEE Transactions on Speech and Audio Processing (To appear)
20. Davies, M., Daudet, L.: Fast sparse subband decomposition using FIRSP. In: Proceedings of the 12th EUropean SIgnal Processing COnference. (2004)
21. Davies, M., Daudet, L.: Sparse audio representations using the MCLT. Signal Processing (to appear)
22. Molla, S., Torrésani, B.: Determining local transientness of audio signals. IEEE Signal Processing Letters **11** (2004)

Dimensionality Reduction in Harmonic Modeling for Music Information Retrieval

Tim Crawford[1], Jeremy Pickens[2], and Geraint Wiggins[1]

[1] Goldsmiths College, University of London,
Centre for Cognition, Computation and Culture
{t.crawford, g.wiggins}@gold.ac.uk
http://www.goldsmiths.ac.uk

[2] King's College, London, Department of Computer Science
jeremy@dcs.kcl.ac.uk
http://www.dcs.kcl.ac.uk/staff/jeremy/

Abstract. A 24-dimensional model for the 'harmonic content' of pieces of music has proved to be remarkably robust in the retrieval of polyphonic queries from a database of polyphonic music in the presence of quite significant noise and errors in either query or database document. We have further found that higher-order (1st- to 3rd-order) models tend to work better for music retrieval than 0th-order ones owing to the richer context they capture. However, there is a serious performance cost due to the large size of such models and the present paper reports on some attempts to reduce dimensionality while retaining the general robustness of the method. We find that some simple reduced-dimensionality models, if their parameter settings are carefully chosen, do indeed perform almost as well as the full 24-dimensional versions. Furthermore, in terms of recall in the top 1000 documents retrieved, we find that a 6-dimensional 2nd-order model gives even better performance than the full model. This represents a potential 64-times reduction in model size and search-time, making it a suitable candidate for filtering a large database as the first stage of a two-stage retrieval system.

1 Introduction

1.1 Harmonic Modeling

A practical system for music information retrieval (MIR) needs to be both effective and efficient. Among the reasonable criteria of effectiveness are robustness to the errors commonly encountered in a musical context, both in queries and the database documents, and to reasonable levels of noise; this robustness needs to be judged against standard evaluation measures such as precision/recall curves. Robustness to error almost always has the effect of harming overall precision since more non-relevant documents are 'recognised' to be similar to the query as higher rates of error are allowed. By using multiple searches in a gradual process of refinement, however, we should be able retain both robustness and precision/recall performance. This would only be possible if our searches were highly efficient, that is, as fast as possible.

In previous work ([1,2,3]) we have described a system for music information retrieval (MIR) which attempts to capture the general harmonic similarity between a musical query and the musical documents sought in a collection. Our simplest, 0th-order model is built by estimating the relative strength of the members of a lexicon of chords, given the observable notes in a window traversing the music, and by consolidating the set of these partial observations (one for each window position) into an overall harmonic signature for the piece, which, like each of the partial observations, is a probability distribution over the lexical chords rather than a single value. This process is applied to all files in the database of music to be searched, and their signatures stored, and at search-time to the query itself; retrieval consists of rating the divergence of the query signature from all of those stored for files in the database. This list is sorted inversely by the divergence, giving a ranked list by harmonic similarity.

The chord-lexicon we use is the 24 major and minor triads. In order to allow for approximate matching between harmonic descriptions, we feel it is essential to find a representation of the harmonic unfolding of the music whereby notes in a query different from those in the original document lend appropriate weight to a matching function according to their harmonic distance from it. There is no music-theoretical way to judge the distance between all possible chords that may arise in the course of a piece of music, but we do have such a measure for the set of 24 triads. Krumhansl and Shepard ([4,5]) present a set of coordinates of relative positions of the triads in a four-dimensional space derived directly from the results of rigorous perceptual testing. We use the simple Euclidean distance between the triads in this space ($Krum(x,y)$ in Equation 1 below) as the basis for constructing our model.

As well as the simple harmonic model described above, we use higher-order models to capture not just the local harmonic description of each window on the music, but also the harmonic transitions between windows. This gives rise to an exponential growth in model size with order, which severely curtails performance in terms of efficiency, although it adds considerable richness to the model.

1.2 Previous Work

Among previous work, that which most closely resembles ours is that by Purwins et al [6]. The authors devised a method of estimating the similarity between two polyphonic audio music pieces by fitting the audio signals to a vector of key signatures using real-valued scores, averaging the score for each key fit across the entire piece, and then comparing the averages between two documents. As do we, these authors use Krumhansl's distance metrics to assist in the scoring. One of the main differences, however, is that these authors attempt to fit an audio source to a 12-element vector of keys, while we fit a symbolic source to a 24-element vector of major and minor triads. Furthermore, by averaging their key-fit vector across the entire piece, their representation is analogous to our 0th-order Markov models. In our work we utilize not only 0th-order models, but 1st and 2nd-order models as well. Moreover, the Purwins paper was not specifically developed as a music retrieval task, and thus has no retrieval-related evaluation.

Shmulevich et al ([7, 8]) also use some of the same techniques presented here, such as Krumhansl's distance metrics and the notion of smoothing. Shmulevich's work is on monophonic music, but demonstrates that harmonic analysis and probabilistic smoothing can be valuable components of a music retrieval system. Other recent techniques which apply 1st-order Markov modeling to monophonic note sequences include [9] and [10]. Further work extends the modeling to the polyphonic domain, using both 0th- and 1st-order Markov models of raw note simultaneities to represent scores [11].

1.3 Our Method

The harmonic models we have developed [1] work by mapping each 12-dimensional note onset vector s onto a 24-dimensional chord vector of the 12 major and 12 minor triads. This ad hoc mapping takes into account not only the size of the overlap between the note and the chord, but also the total number of notes in the simultaneity, and the Krumhansl and Shepard ([4, 5]) perceptual distance between the chords in the lexicon:

$$Context(s,c) = \frac{|s \cap c|}{|s|} \sum_{c' \in lexicon} \frac{|s \cap c'|}{(|s| \times Krum(c',c)) + 1} \tag{1}$$

This context score is computed for every chord c in the lexicon. Additionally, inter-vector smoothing[1] is performed, whereby neighboring vectors are allowed to contribute to the partial observation of the current vector. A vector of partial observations is then obtained by normalizing by the sum total:

$$PartialObs(s,c) = \frac{Context(s,c)}{\sum_{c' \in lexicon} Context(s,c')} \tag{2}$$

This vector of partial observations over the chord lexicon is then used as the raw feature set for model estimation. For example, suppose we have a lexicon of three chords, P, Q, and R. A sequence of partial observation vectors might appear as in Table 1.

We simply count the number of length m sequences through a piece of music, each count weighted by the fractional observation amount. Suppose $m = 2$. We begin with the window from timestep 1 to timestep 2. The sequence $P \Rightarrow P$ is observed in proportion to the amount in which we observe P at timestep 1 and also observe P at timestep 2 ($0.2 \times 0.1 = 0.02$). The sequence $Q \Rightarrow R$ is observed in proportion to the amount in which we observe Q at timestep 1 and then R at timestep 2 ($0.5 \times 0.8 = 0.4$), and so on.

We next divide these chains into two parts, the previous state, or history, and the current state. We define the history H as the first $m - 1$ chords in the sequence, and the current state c as the final chord in the sequence. For example, with an $m = 2$ chain "$P \Rightarrow Q$", the history is the state "P" and the current state is "Q". With an $m = 3$ chain "$P \Rightarrow Q \Rightarrow P$", the history is the state "$P \Rightarrow Q$" and the current state is "P".

[1] Note that we are smoothing the data *before* the model is generated, rather than smoothing the model itself, as is the common practice.

Table 1. Partial observation vectors

	Partial observation vectors				
Chord	1	2	3	4	5
P	0.2	0.1	0.7	0.5	0.0
Q	0.5	0.1	0.1	0.5	0.1
R	0.3	0.8	0.2	0.0	0.9
Total	1.0	1.0	1.0	1.0	1.0

We ask the question: "Given that I have just seen the history H, what is the probability of seeing c as my current state?". We answer this question by doing maximum likelihood estimation. From our training data we count the number of times we see the full sequence "$H \Rightarrow c$", and divide by the number of times we see the sequence $H \Rightarrow x$, where x represents any chord. This process yields a estimate of the conditional probability distribution $\dot{P}(c|H)$.

Prior to retrieval, at indexing time, we estimate $\dot{P}(c|H)$ for every piece of music in the collection. At retrieval time, when presented with a query, we estimate a model for the query in the exact same manner. Similarity is calculated between the query model and every document model in the collection using an appropriate divergence measure. The documents in the database are ranked in increasing order of divergence from the query.

1.4 Size and Performance of Harmonic Models

The harmonic models generated for each database document, especially those of higher order, can become very large. Reducing the dimensionality brings obvious benefits in terms of model size, and thus in search time.

For a practical MIR system based on this method, it is clear that we need to find ways to reduce the dimensionality of our models. But it is also desirable to retain the perceptual/cognitive musical basis for the model so we can still understand the process of matching, at least to some extent. For this reason, our first steps in this direction are not based on statistical or other mathematical dimension-reduction techniques, but on more straightforward means of simplifying the harmonic models.

To assess the impact of dimensionality reduction on overall performance we conducted our experiments using a version of the Cranfield IR evaluation criteria ([12, 13]), familiar from the TREC series of conferences [14], which are being largely adopted by the MIR community. At present, working within a prototype experimental framework rather than a development environment, we are not concerned with performance speed, per se, and merely seek to achieve smaller models while maintaining the highest possible precision/recall figures. We expect to see a reduction in this performance measure with decreasing dimensionality, and that is borne out in the results below.

The Cranfield/TREC evaluation paradigm depends on the existence of standard data-sets from which documents will be retrieved, a set of standard query tasks, and a corresponding list of relevance judgments; that is, a list of documents judged by human experts to be relevant to the standard queries.

Mention of 'relevance' here inevitably raises the problematic question as to what is meant by the term in a musical context [15]. In [16] it was suggested that a musical query could be judged to be relevant to a database music document according to how well it 'evokes' that document. We believe there is one category of musical form that depends crucially on this quality of 'evocation': the Theme and Variations [17]. Almost by definition, a theme is mutually relevant to the variations. However, this relevance may be 'evoked' in different ways by a composer in different variations from a single set and these ways will differ at different periods of music history. As our matching technique is founded upon the notion of common harmonic makeup between query and document, it seems reasonable, therefore, to restrict ourselves to choosing sets of variations which are built on a roughly-invariant harmonic structure, although we admit varying degrees of exception to this which will be outlined below. This is generally the case with composed variation-sets from the 18th century or before.

We feel that recognizing variations on a given theme (or its converse, identifying a theme from which a variation was derived) is such a fundamental musicological task that a system that carries it out well will be warmly welcomed. Furthermore, variations need not necessarily be 'composed'; they arise naturally (albeit on a finer-grained scale) when different performances of the same work are recorded, and particularly when different artists cover the same popular song in differing arrangements. In the last case, generally speaking, while in surface detail performances may differ greatly, the almost-invariant harmonic underpinning of popular music of all periods is very marked, although we do not test this assertion in these experiments. (We would not make such a strong statement about jazz, however; this is a genre in which the substitution of chords and chord-sequences, which—although they are basically closely-related—are often very different in different interpretations of a 'standard', plays an important part in the creative process.)

Although the present paper does not deal with the matching of 'real-world' audio recordings, we should like to draw attention to the fact that our basic harmonic-matching system has already been used successfully for retrieval of audio queries from a symbolic database [3, 2], and, more recently, for retrieving audio queries from an database of cover versions of audio pop and jazz recordings [18].

We envisage using lower-dimensionality models in a working MIR system in two basic ways: a) as a means to 'filter' unlikely matches from a large database, and b) as a first step towards an effective indexing technique for polyphonic music. A third possibility is to combine the results from multiple low-dimension searches in ways that work to enhance precision/recall.

2 Dimensionality Reduction in Harmonic Models

Dimensionality reduction can be done in very many different ways. These first experiments adopt a straightforward approach in order to assess the basic practicality of the method. We can derive n-dimensional models directly from the 24-d ones by either simply selecting n dimensions from the 24 and ignoring the rest, or by 'binning' adjacent clusters of $\frac{24}{n}$ neighboring dimensions into n bins.

Our harmonic-modeling method is based on the table of four-dimensional inter-triad distances obtained from music-psychological experiments reported in [5]; this can be projected into two dimensions as a pair of interleaved 'circles of fifths', one for each of the major and minor triads (see Figure 1). Our models are distributions across a chosen lexicon of triads; the size of this lexicon defines the dimensionality of our models. For our earlier work we used the full set of 24; in these experiments we shall be using either i) a principled subset of triads or ii) agglomerations of neighbouring triads. In the first case, we have various options; in the work reported here, we simply select a) all 12 major triads (see Fig. 1a) and b) all 12 minor triads. In the second case we construct models of 12, 8, 6 and 4 dimensions by summing groups of 2, 3 (see Fig. 1b), 4 and 6 neighboring triads from the circle and re-normalizing the resulting distributions.

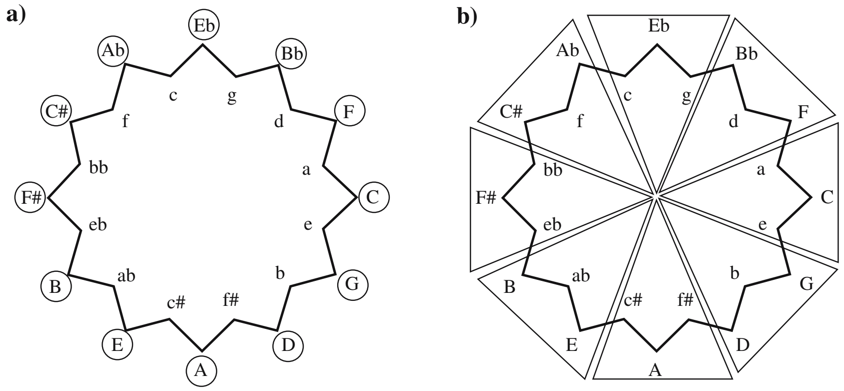


Fig. 1. Dimensionality might be reduced by a) selecting certain dimensions, in this case those corresponding to the major triads only, or b) by combining dimension-values corresponding to neighboring triads on the double-circle of fifths, in this case bins of three values resulting in an 8-dimensional model

As stated above, we feel it is important to retain as much 'intuitive' grasp of what is happening in our matching mechanism, and these simple dimensionality-reduction methods offer that possibility. However, it must be said that we cannot predict with any certainty whether or not any method of dimensionality reduction will necessarily and inevitably harm the overall retrieval performance of an MIR system using harmonic modeling for all queries and all databases. Generally speaking, we feel that it is not often realized how sensitive almost any musically-principled data model for MIR is likely to be to the actual nature of the music in the database and the queries themselves. We have found, for example, not altogether surprisingly, that sets of music documents judged to be 'relevant' to a certain set of queries (i.e. as sought by a user running those queries) are likely to lead to very different retrieval results with different models depending on whether or not the relevant documents are or are not in the same key as a query. Another issue is the more subtle question of the musical 'texture' of a query compared with the database documents sought by the user; whether, for example, the music is largely melodic or chordal, and in the latter case whether it is highly arpeggiated or consists of block chords.

In fact, we believe that in general in MIR, any results are likely to depend very much on the nature of the queries, the relevance judgements and the database itself. The only way to ensure that certain databases are suitable for specific retrieval tasks will be to carry out exhaustive user testing. As yet, we have little idea of how many generic music-retrieval tasks exist, which is another area of user-based research that needs further effort.

Also the nature of the retrieval task is important: if we are intending to use low-dimensionality models in an initial filtering stage of a general MIR system before we proceed to a detailed passage-level search using some other method on a reduced number of documents from the database, for example, then high recall is probably more important than high precision. [19] For a task explicitly based on matching harmonic structure in the music, high precision possibly would be a higher priority.

3 Experiments

In order to test the precision/recall performance of a system based on the new reduced-dimensionality models described here, we needed to built a standard test set following the TREC model, which comprises a database of music documents in which the searches will take place and a set of music queries with a set of associated judgements as to which documents in the database are 'relevant' to each query. This test set needs to be of reasonable size — the database must be neither so small as to be entirely unrepresentative, nor so large as to cause serious problems due to difficulties with scaling. It would, of course, be useful in other testing contexts.

The search method we use here is identical with that described above and reported previously [1, 3, 2], except that, instead of the widely-used Kullback-Leibler divergence measure for computing (inverse) similarity between query and document models, we use here the recently-reported symmetrical Endres-Schindelin measure [20][2].

3.1 The Standard Database and the Sets of Variations

The standard database we use throughout comprises a collection of nearly 5,500 music files, partly MIDI files in all genres downloaded from the world-wide web,

[2] We would like to thank Tak-Shing Chan (Goldsmiths College) for bringing this new measure to our attention, and we plan to report in detail on our reasons for using it in due course.

and including a large portion of the CCARH MuseData collection [21], together with a number of files of lute music originating in the ECOLM project [22]. It also includes copies of all the music files used as queries in these experiments (duly listed as relevant to the queries); in each case we expect to retrieve these files as first item in the ranked list output by our system. Since this is a uniform result across all experiments it does not affect our findings, which are entirely concerned with comparative performance.

The sets of queries we used, six in total, are of two basic types: a) composed sets of variations (four sets), and b) sets of pieces that are essentially in each case 'variants' of the same musical work (two sets). These are described in more detail below. For each set, each variation is considered 'relevant' to all the others, regardless of any actual differences perceived in listening. In some cases, for example, a relevant file may be in a different key, or mode (major or minor), from its theme; disregarding such anomalies preserves the objectivity of the experiments, and does not affect their validity as a comparative test of the dimensionality-reduction method.

3.2 Query-Sets

The four composed sets of variations were as follows:

1. G.F. Handel's 'Harmonious Blacksmith' variations in E major for harpsichord; two different MIDI performances both in complete versions and segmented into separate theme and variations (total: 13 queries[3]);
2. Handel's Air and Variations in B flat major for harpsichord; a single MIDI performance in a complete version and segmented into separate theme and variations (total: 7 queries);
3. J.S. Bach's Aria and 30 variations in G major, BWV 988, the 'Goldberg' variations, for harpsichord (31 queries; three variations, numbers 15, 21 and 25 are in fact in G minor);
4. Bach's chorale variations on "Sei gegrüßet, Jesu gütig" in G minor, BWV 768, for organ (12 queries).

The two other sets of queries were not composed sets of variations, but were assembled manually:

5. 75 settings (c1590-1670) of John Dowland's "Lachrimae Pavan" in various keys and various scorings (solo lute, keyboard, instrumental consort, solo voice with accompaniment, etc.) gathered and encoded as part of the ECOLM project (75 queries);
6. A collection of versions in various keys of Gounod's "Ave Maria", originally composed as a "Meditation" on the first prelude in C major from Bach's 48 Preludes and Fugues for keyboard, to which Gounod added a vocal part; the set also includes MIDI files of a few versions of Bach's original prelude, one or two of which include the fugue as well (12 queries).

[3] The segmentation of the queries was done slightly differently for the two sets; hence the odd total number of queries.

4 Results and Evaluation

4.1 Retrieval Evaluation

Sets of database models were prepared with dimensionality 24, 12, 8, 6 and 4. In each case we used a Markov chain-length of 3 (corresponding to a 2nd-order model) and an event-based smoothing window of 4 adjacent events; we have found in the course of extended testing that this is the best-performing configuration of these parameters for recognizing variations. Searches were conducted on all six sets of variations, using each theme and all the variations as queries in turn, and treating each of them as relevant documents for the search; the total number of queries executed was thus 150. The results of the searches for each reduced-dimensionality model were averaged over all sets of queries, and standard 11-point interpolated precision/recall figures for the first 1000 documents retrieved extracted from the resulting ranked lists. These were compared and subjected to the t-test for statistical significance.

Table 2. Analysis of retrieval results averaged over all six sets of queries. (Statistical significance in the percentage change from the 24-dimensional baseline as verified by the *t*-test is indicated by an asterisk.)

Dimensions:	**24**	**12**		**8**		**6**		**4**	
			% change		% change		% change		% change
Relevant retrieved:	5646	5983	5.97*	5905	4.59*	6646	17.71*	5398	-4.93*
Interpolated Recall — Precision									
at 0.00	0.9471	***0.9978***	***5.3****	0.9978	5.3*	1.0000	5.6*	0.7572	-20.1*
at 0.10	0.6003	***0.5895***	***-1.8***	0.4882	-18.7*	0.4992	-16.8*	0.3775	-37.1*
at 0.20	0.4660	***0.4648***	***-0.3***	0.3809	-18.3*	0.3674	-21.1*	0.2831	-39.2*
at 0.30	0.4096	***0.3952***	***-3.5***	0.3286	-19.8*	0.3330	-18.7*	0.2254	-45.0*
at 0.40	0.3386	***0.3227***	***-4.7***	0.2809	-17.0*	0.3008	-11.2	0.1835	-45.8*
at 0.50	0.2755	***0.2814***	***2.1***	0.2474	-10.2	0.2651	-3.8	0.14757	-47.1*
at 0.60	0.1311	***0.1572***	***19.9***	0.1435	9.4	0.1060	-19.2	0.0848	-35.4*
at 0.70	0.1173	***0.1141***	***-2.7***	0.0979	-16.5	0.0738	-37.1*	0.0674	-42.5*
at 0.80	0.0733	***0.0872***	***18.9***	0.0848	15.6	0.0541	-26.2*	0.0439	-40.1*
at 0.90	0.0389	***0.0456***	***17.0***	0.0466	19.7	0.0141	-63.8*	0.0126	-67.6*
at 1.00	0.0162	***0.0244***	***50.5****	0.0294	81.3*	0.0117	-28.1*	0.0073	-54.8*
Average precision (non-interpolated) over all relevant documents									
	0.2835	***0.2843***	***0.28***	0.2438	-14.01*	0.2334	-17.67*	0.1649	-41.83*
Precision:									
at 5 docs:	0.6893	0.6213	-9.9*	0.5133	-25.5*	0.3933	-42.9*	0.4653	-32.5*
at 10 docs:	0.5713	0.5320	-6.9*	0.4127	-27.8*	0.3333	-41.7*	0.3967	-30.6*
at 15 docs:	0.5107	0.4796	-6.1*	0.3667	-28.2*	0.3422	-33.0*	0.3587	-29.8*
at 20 docs:	0.4693	0.4427	-5.7*	0.3510	-25.2*	0.3430	-26.9*	0.3313	-29.4*
at 30 docs:	0.4033	0.3956	-1.9	0.3216	-20.3*	0.3373	-16.4*	0.2849	-29.4*
at 100 docs:	0.2427	0.2325	-4.2*	0.2101	-13.4*	0.2525	4.0	0.1681	-30.7*
at 200 docs:	0.1479	0.1488	0.6	0.1372	-7.2*	0.1540	4.2*	0.1157	-21.7*
at 300 docs:	0.0681	0.0705	3.6*	0.0681	0.0	0.0715	5.0*	0.0617	-9.4*
at 1000 docs:	0.0376	0.0399	6.0*	0.0394	4.6*	0.0443	17.7*	0.0360	-4.4*

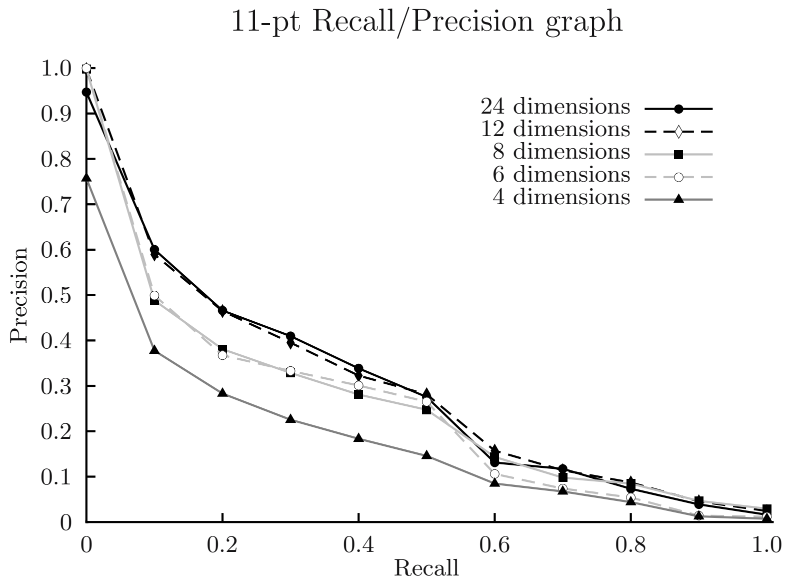


Fig. 2. Recall/precision curves for each model-dimensionality, averaged over all queries

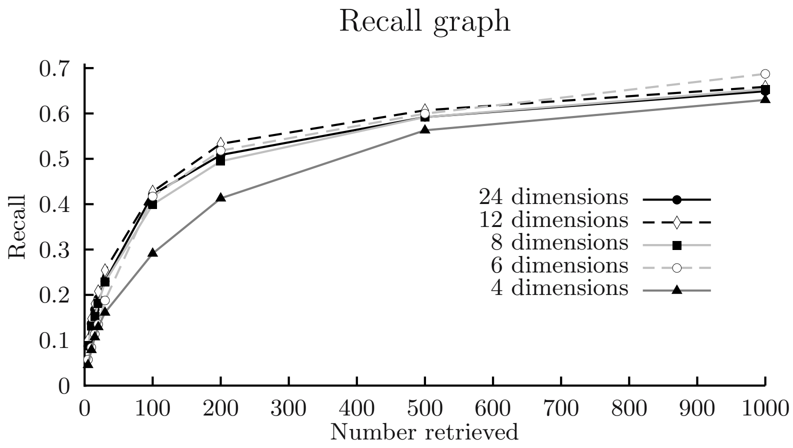


Fig. 3. Recall against number of documents retrieved for each model-dimensionality, averaged over all queries

The total number of documents retrieved in the first 1000 for each search over all queries was 150000; the total number of relevant documents was 11340. A statistical summary of the retrieval results is given in Table 2. The 11-point interpolated recall/precision curves are shown in Fig. 2, and a plot of recall against number of documents retrieved is given as Fig. 3.

Considering precision first, from Fig. 2 we can observe a general and expected reduction of precision as model dimensionality is reduced, although this is not a simple linear decline. However, as the percentage change figures given in Table 2 show, the difference between the recall/precision performance of our 24-dimensional and 12-dimensional models is not statistically significant. (This group of figures is given in bold and italic in the table.) Turning to the recall curves plotted in Figure 3, we can see that the total number of relevant documents retrieved for each search does not necessarily decline with decreasing dimensionality. In fact, at 1000 documents, our 6-dimensional model is by far the best performer by this criterion (see the figures shaded in Table 2), returning 17.7% more relevant documents on average.

5 Discussion

The most important result from our experiments is that we can indeed achieve equivalent precision/recall retrieval effectiveness from a 12-dimensional model to that we can from our full 24-dimensional one. This will have a very significant impact on retrieval efficiency as well, before we consider other speed-up techniques, such as indexing.

This improvement in efficiency can be easily demonstrated by considering the basic number of parameters in our models at different dimensions. The determining factor for the model size, that is, the number of parameters, n, of a model of dimensionality d and Markov order m is given by the simple formula: $n = d^{(m+1)}$.

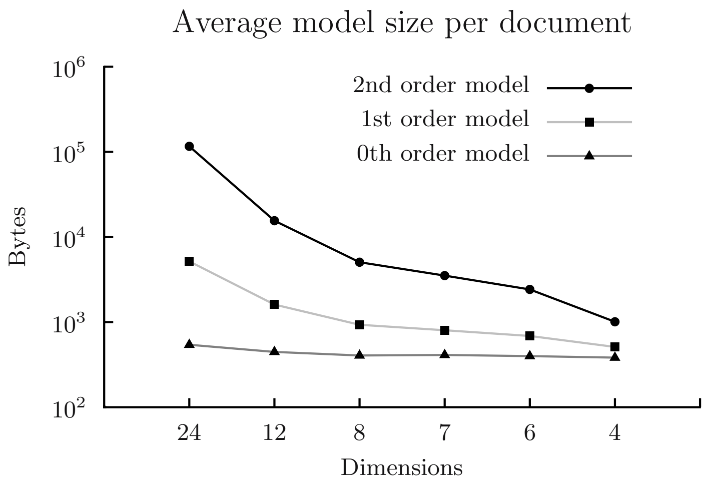
Thus for the 2nd-order models used in these experiments, the model size decreases with dimensionality according to Table 3:

Table 3. Decrease in 2nd-order model size with number of dimensions

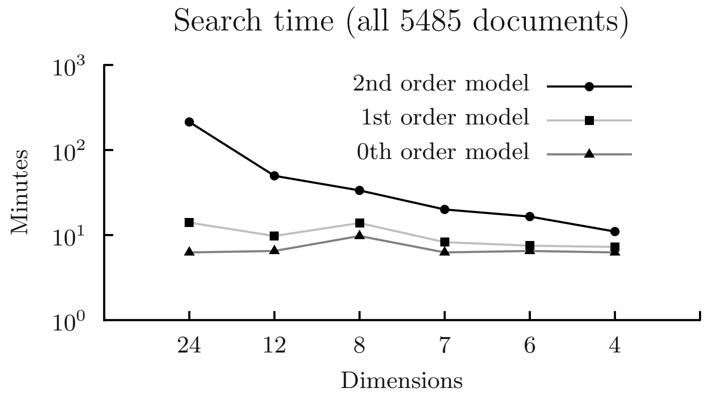
Dimensions	n
24	$24^{(2+1)} = 13,284$
12	$12^{(2+1)} = 1,728$
8	$8^{(2+1)} = 512$
6	$6^{(2+1)} = 216$
4	$4^{(2+1)} = 64$

Firstly, our 12-dimensional models are 8 times smaller than the 24-dimensional ones, and we can thus expect an eight-fold reduction in search-time by using them; as we have seen from the recall/precision results, this is at no statistically-significant cost in precision.

Secondly, our 6-dimensional models are another 8 times smaller again, so with these models we can expect a decrease in search time of no less than 64 times compared with the 24-dimensional models. While we have seen that the precision of these 6-dimensional-model searches will not be as good as with the 24-dimensional models, their recall is much better.



(a) Average harmonic-model size per document in bytes (log scale)



(b) Elapsed time in minutes (log scale)

Fig. 4. Size and time results for a typical retrieval experiment (75 queries) plotted against dimensionality of model

Any reduction we can achieve will have a beneficial effect on retrieval response time in particular, as larger models take longer to match. Figure 4 demonstrates informally how model size and search times decrease dramatically as dimensionality is reduced in practical tests using the methods described in this paper.

Given these improvements in efficiency, we could in fact carry out multiple low-dimensional searches in much less time than a single 24-dimensional search. An obvious method for retrieval will be to use the fastest, 6-dimensional-model search as an initial filter, passing the top 1000 retrieved documents to a second stage where they are searched with much greater precision using a 12-dimensional model.

Some documents will, of course, not be retrieved at all in this two-stage process: those that do not appear within the top 1000 documents for the 6-dimensional-model search. While this may at first sight seem to be a problem, we are not very concerned about it. The reason is that, actually, the retrieval task we have set ourselves — to retrieve all variations in each of the query-sets — is not likely to be achievable with our present simple models on two counts. Firstly, some of the queries are in different keys from the others (see query sets 5 and 6), and some of the queries are in a different mode (major or minor) from the others (three of those in query set 3). These 'anomalous' queries typically appear well below the 1000-document threshold in retrieval experiments. We do, however, have methods for dealing with these particular problems, which we are actively pursuing.

6 Further Work

6.1 Other Reduced-Dimensionality Models

It is possible to devise other reduced-dimensionality models of harmony, in fact, an indefinite number of them. We have restricted ourselves to those for which we have some intuitive grasp of the way they probably work. Reducing dimensions by statistical methods, or by more complex selection of dimensions, is likely to remove this intuitive aspect. However, we have tested two 7-dimensional models whose dimensions are based on triads built on the seven members of the C major scale (C, D, E, F, G, A, B), and on those of the G minor natural scale (G, A, B♭, C, D, E♭, F). With our test data, these performed extremely well, but we do not present these results here, as we feel that they are probably biased by the tonality of the query-sets and are unlikely to represent any kind of universal performance. Furthermore, we find it hard to explain exactly which aspects of harmony we are in fact modeling with them. It would be possible to reduce model dimensionality by selecting dimensions according to the tonal/harmonic makeup of the database, but this would have to be accompanied by a great deal of testing. The same would be true of statistical dimensionality-reduction methods, but they would have the advantage of objectivity.

6.2 Indexing

We hope that the reduced-dimensionality models can be used as a basis for an indexing strategy. Multi-dimensional indexing is a complex area, with many applications, and we therefore hope that general solutions will emerge. However, it seems likely that the fewer dimensions are involved the more successful any indexing system will be. Perhaps the simplest method would be to decide on a number of harmonic states into which we bin our harmonic signatures. This is effectively a reduction to a single dimension of harmonic labels which can be searched very quickly. This would actually be most powerful as a method for passage-level retrieval, where we would produce a string of such harmonic labels for each piece from our partial observation vectors rather than the complete-piece signatures. Some such data-structure as the suffix tree could then be used for very efficient searching[4].

6.3 Smoothing

In earlier work, we showed the importance of smoothing, in essence, the choice of size of the window with which we traverse the musical data and collect the pitch data from which we build our harmonic partial observations. Effectively, this is a process of shifting some of our partial observations forward in time, to reduce the problems caused by gaps, deletions and insertions in polyphonic music; this is the main reason for the general robustness of our matching method. We have consistently found that, except with 0th-order models, smoothing is essential to achieve reasonable retrieval results, and, furthermore, that larger amounts of smoothing are more beneficial as model order increases. In our present work we use naïve, event-based smoothing, just as we did in our earlier experiments. It is quite possible that a more 'intelligent' smoothing mechanism, either based on onset-time or beat segmentation, or some form of adaptive method, would give better retrieval results, but this needs to be the focus of further research. The crucial point is that the smoothing of queries and documents must be consistent in a search.

6.4 Transposition Independence

In [1] we describe a transposition-invariant version of our model, which allows us to match queries and documents containing similar harmonic transitions, but globally transposed onto a different pitch-level or key. An inevitable consequence of allowing these matches is some general loss of precision in the un-transposed matching task. Although we have not conducted a full set of tests at the time of writing, there is some evidence that this loss of precision is in fact less in some lower-dimensional models than in the full 24-dimensional ones. More testing is needed, but if this initial finding is confirmed, it is another very positive outcome of the dimensionality-reduction exercise that we hope to exploit in future work.

6.5 Merging Results

It is a well-known result that if multiple classifiers each give results better than random, one can achieve results better than each classifier individually by combining their classification hypotheses. [24] We can treat our ranked-list results as classification hypotheses, and 'merge' the lists to obtain a more effective ranking. We are conducting tests with ranked lists output by searching with different models, and hope that this could be a means to enhance recall/precision performance.

[4] This technique has been used by Michael Casey in an audio-retrieval system [23].

Acknowledgements

The work that is reported here is an extension of previous research carried out as part of the OMRAS (Online Music Recognition And Searching) project at Queen Mary College, King's College and Goldsmiths College, University of London, in the U.K. and the University of Massachusetts at Amherst and Indiana University, in the U.S. [25]. The initial project (1999-2003) was supported by grants from the Joint International Digital Libraries Initiative funded by the JISC and NSF, respectively. The current work is supported by EPSRC grants GR/N63123/01 (King's College) and GR/S17253/02 (Goldsmiths College). We are grateful to Alastair Craft for assistance in the preparation of this paper.

References

1. Pickens, J., Crawford, T.: Harmonic models for polyphonic music retrieval. In: Proceedings of the ACM Conference in Information Knowledge and Management (CIKM), McLean, Virginia (November 2002)
2. Pickens, J., Bello, J.P., Monti, G., Crawford, T., Dovey, M., Sandler, M., Byrd, D.: Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. Journal of New Music Research **32** (2003) 223–226
3. Pickens, J., Bello, J.P., Monti, G., Crawford, T., Dovey, M., Sandler, M., Byrd, D.: Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. In: Proceedings of 3rd International Symposium in Music Information Retrieval (ISMIR), IRCAM, Paris (2002) 140–149
4. Krumhansl, C.L., Shepard, R.N.: Quantification of the hierarchy of tonal functions within a diatonic context. Journal of Experimental Psychology: Human Perception and Performance **5** (1979) 579–594
5. Krumhansl, C.: Cognitive Foundations of Musical Pitch. Oxford University Press, New York (1990)
6. Purwins, H., Blankertz, B., Obermayer, K.: A new method for tracking modulations in tonal music in audio data format. In Amari, S., Giles, C., Gori, M., Piuri, V., eds.: International Joint Conference on Neural Networks, IJCNN 2000. Volume 6. (2000) 270–275 `http://doi.ieeecomputersociety.org/10.1109/IJCNN.2000.859408.`
7. Shmulevich, I., Yli-Harja, O., Coyle, E., Povel, D., Lemström, K.: Perceptual issues in music pattern recognition — complexity of rhythm and key find. Computers in the Humanities **35** (2001) 23–35
8. Shmulevich, I., Yli-Harja, O., Coyle, E., Povel, D., Lemström, K.: Perceptual issues in music pattern recognition — complexity of rhythm and key find. In: Proceedings of the AISB99 Symposium on Musical Creativity, Florida. (1999)
9. Rand, W., Birmingham, W.: Statistical analysis in music information retrieval. In: Proceedings of the 2nd International Symposium on Music Information Retrieval, Indiana University, Bloomington, Indiana (2001) 25–26
10. Hoos, H.H., Renz, K., Görg, M.: Guido/mir — an experimental music information retrieval system based on guido music notation. In: Proceedings of the 2nd International Symposium on Music Information Retrieval, Indiana University, Bloomington, Indiana (2001) 41–50

11. Birmingham, W., Dannenberg, R.B., Wakefield, G.H., Bartsch, M., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M., Rand, M.: Musart: Music retrieval via aural queries. In: Proceedings of the 2nd International Symposium on Music Information Retrieval, Indiana University, Bloomington, Indiana (2001) 73–81
12. Cleverdon, C.W., Mills, J., Keen, M.: Factors determining the performance of indexing systems; volume 1, design. Technical report, ASLIB Cranfield Project, Cranfield University, Cranfield, UK (1966) `http://hdl.handle.net/1826/861`, `http://hdl.handle.net/1826/862`.
13. Cleverdon, C.W., Keen, M.: Factors determining the performance of indexing systems; volume 2, test results. Technical report, ASLIB Cranfield Project, Cranfield University, Cranfield, UK (1966) `http://hdl.handle.net/1826/863`.
14. TREC: Text retrieval conference. (`http://trec.nist.gov/`)
15. Byrd, D., Crawford, T.: Problems of music information retrieval in the real world. Information Processing and Management **38** (2002) 249–272
16. Pickens, J.: Harmonic Modeling for Polyphonic Music Retrieval. PhD thesis, University of Massachusetts at Amherst (2004)
17. Sisman, E.: Variations. `http://www.grovemusic.com` (2005) Accessed 25 July, 2005.
18. Bello, J.P., Pickens, J.: A robust mid-level representation for harmonic content in music signals. In: Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2005, Queen Mary College, University of London (September 2005)
19. Downie, J.S.: Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-grams as Text. PhD thesis, University of Illinois at Urbana Champaign (1999)
20. Endres, D.M., Schindelin, J.E.: A new metric for probability distributions. IEEE Transactions on Information Theory **49** (2003) 1858–1860
21. Center for Computer-Assisted Research in the Humanities, Stanford University (CCARH): Musedata collection of encoded scores. (`http://www.musedata.org`)
22. ECOLM: Electronic corpus of lute music. (`http://www.ecolm.org`)
23. Casey, M.A.: Acoustic lexemes for organizing internet audio. Contemporary Music Review ((accepted for publication) 2005)
24. Pickens, J.: Classifier combination for capturing musical variation. In: Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2005, Queen Mary College, University of London (September 2005)
25. OMRAS: Online musical recognition and searching. (`http://www.omras.org`)

Abstracting Musical Queries: Towards a Musicologist's Workbench

David Lewis, Tim Crawford, Geraint Wiggins, and Michael Gale

Centre for Cognition, Computation and Culture,
Goldsmiths College, University of London,
New Cross Gate, London SE14 6NW, UK
{d.lewis, t.crawford, g.wiggins, m.gale}@gold.ac.uk

Abstract. In this paper, we propose a paradigm for computer-based music retrieval and analysis systems that employs one or more explicit abstraction layers between the user and corpus– and representation–specific tools. With illustrations drawn from "battle music", a genre popular throughout Renaissance Europe, we show how such an approach may not only be more obviously useful to a user, but also offer extra power through the ability to generalise classes of tasks across collections.

1 Introduction

As more digital resources are created for musicologists and musicians, there is an increasing demand for tools that can facilitate the use of these corpora and make them accessible for both retrieval and analysis. There are several impediments to the development of such tools, the most significant of which arise from the diversity of both the potential user base and the nature and structure of the resources themselves. In order to construct general tools for a variety of users and a range of digital musical resources, we need to define a domain of interaction between user and data that can be abstracted both from the electronic data and also from the users' descriptions of music.

A generalized system for music retrieval and analysis requires several elements. Firstly, it needs an ontology[1] which will provide methods for describing musical features observed or sought by users and which may be mapped, with reasonable fidelity, on to descriptions of the features as they occur within verbalised musical discourse. Secondly, it needs a structural definition of how to combine these ontological elements and have them interact to form a description of a feature-based query. Finally, it needs interfaces between users and this layer of abstraction, and between it and the music data: the former tailored to users' interests and requirements, and the latter to the special nature of the data in a corpus, its formats and its representations. Figure 1 provides a schematic representation of such a system.

Such a system gains descriptive power – and generality – by separating the user from the implementation details of the corpora and the methods used to

[1] Our use of the term follows [1].

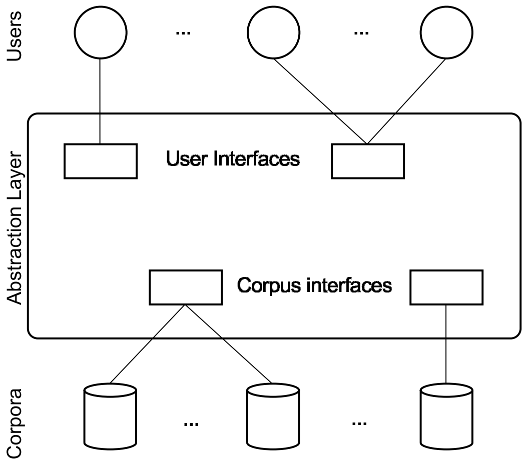


Fig. 1. Schematic representation of a system in which the users' interactions with corpora are always mediated by an abstraction layer

retrieve information from them. The users' communities and the areas of their studies will inform their needs, and their modes of interaction with a Music Information Retrieval (MIR) system should vary accordingly. Similarly, the individual needs of a corpus, its medium or its representation format create a parallel need for abstraction at a lower level.

Whilst the low-level corpus interface has received limited attention [2, 3, 4, 5], the higher-level interface has received almost none (exceptions are [6] and [7], both of which point to the need for such an abstraction for use by musicologists, although little further detail is given in either case).

In this paper, we introduce elements of this abstraction layer, discussing and illustrating some of the properties the layer and its components will require. We conclude by expanding our scope to include metadata in its broadest sense and considering how to move towards a fuller specification for our system.

2 Definitions

Before we can discuss the vocabularies of users, we must first explain our own, since we use a number of terms in an altered or specialized sense.

Feature. In a musical sense, any aspect of a piece that may be described, and located from that description (see, for example, the use of the term in [8]). Usually the aspect will be melodic, rhythmic, textural, timbral or any combination of these.

Simple Feature. A feature that cannot usefully be subdivided into a combination of smaller features. A single note duration or interval, for example, would be considered to be a *simple feature* in this sense.

Compound Feature. A feature that is recognizable as being made up of several features in combination. *Compound features* are usually specific and named in our usage, serving to mark out individuating aspects of a class of pieces, whereas a *simple feature* may be ubiquitous, and therefore not discriminatory.

Constituent Feature. A *feature* that forms part of a *compound feature.*

Feature Description. Where *feature* is roughly analogous to a part of a user need (that is, a musical idea sought in a corpus), the *feature description* is part of the user narrative (an expression of that need or description of the idea), on its way to becoming a formalised query.

3 Towards a Vocabulary for Musicological Queries

High-level verbal descriptions of music tend to function primarily in two ways: on the one hand, they may draw attention to elements of a work or group of works that resemble others, often implying some degree of kinship; and on the other, they may show elements that distinguish one work or group from another.[2] In either case, the description not only requires knowledge of the music that is being explicitly described, but also draws upon a reference set – or repertory – with which all comparisons are implicitly made. This repertory might consist of a list of specific works, or even simply a genre or period – such as "Romantic opera", "1940s swing" or "Renaissance lute music" – about which a degree of familiarity can be assumed.

Each repertory has a community around it who will take it, and how it is constituted, for granted. The community fashions its own vocabulary for discussing the repertory and relating other music to it. This vocabulary may use different, specialist terms for some elements of the music or may simply apply existing terms in novel or more specific ways.[3] Not only do some elements of the vocabulary mean different things within different communities, but the same feature may be described in many different ways. It is clear, then, that it will be necessary to explore different communities and their respective discourses to get a strong idea of how these vocabularies may be applied in a computational context. Such a broad investigation has not yet been carried out.

In the context of MIR, a few surveys (see, for example, [11] and [12]) have investigated the questions that people ask – or would like to be able to ask – of music resources, physical or virtual. Such surveys greatly enhance our understanding of users' vocabularies, methods and practices, but they are usually drawn from a single amalgamated sample or are divided into groups by social rather than musical criteria. By using questions that relate clearly to current applications of MIR or by basing data on existing querying practices, studies

[2] This distinction is visible in the division traditionally drawn between music theory and music analysis.[9]

[3] Here, our concept of *community* is related to that described by Kuhn in his postscript to [10].

have tended to limit participants' contributions so that only existing information retrieval paradigms are explored. The inevitable ambiguity that arises from the use of multiple vocabularies may be greatly reduced by separating a user interface (which may be customized for a single community) from the more generalised abstraction layer that articulates the query to the search tools. Those search tools can, in turn, be specialized to the material on which they work, allowing sensitivity to the nature of the data.

The machine-representation side of data abstraction at fine-grained levels of detail has been studied elsewhere [2, 3]; to achieve the ends sought here, higher level structures and predications about them would be necessary [4, 5]. However, the focus of the current paper is on the musical side of the abstraction layer, and therefore we discuss the machine side no further.

3.1 Example: 'Battle' Pieces

To illustrate our approach and demonstrate its relevance and usefulness for current musicology, we look at a recent study by Michael Gale [13] of a loosely-defined group of 'battle' pieces which date from the early sixteenth century into the middle of the seventeenth [14]. In this example, our repertory (in the sense described above) is European music of the Renaissance, taking in vocal genres, ensemble music and keyboard and lute traditions. The vocabulary we generate for talking about these pieces, then, will be constructed in relation to this background, but we shall also look at how its elements might contribute to a more general ontology.

Battle pieces vary greatly in form, instrumentation, length and function: their similarity arises primarily from the deployment of features drawn from a common palette – with no requirement that they be consistently used in their original context. These features may be melodic (often quoting from popular songs), textural or harmonic in nature, and it is this variety that makes them of particular interest in the context of this paper.

The spectacular popularity of these pieces in their own time means that they are extremely widespread across surviving music sources: hundreds of them appear in music manuscripts and prints from the period. This means that the retrieval task of distinguishing these widely-dispersed pieces in a larger encoded corpus is one with real applications. Furthermore, Gale [13] discerns subgroups of battles, each distinguished to a varying degree by the presence of its own additional features or its particular use of the general battle features.

Gale and Lewis [15] implemented search tools to find several of the features described, operating on a mixed-format corpus of MIDI and TabCode[4] files. They presented the results to illustrate the efficacy of searching a corpus by combining several different features; here, we concern ourselves more with the general attributes of the features themselves and how they combine and interact.

An example of how compound features may be produced through the combination of smaller elements may be seen in a common feature, originating from a

[4] An ASCII encoding scheme for lute tablature. [16]

passage in the most influential battle piece of all, Janequin's chanson *La Guerre*, and appearing in a substantial number of subsequent works. Figure 2 shows this feature as it appears in the original chanson (a) and in a later instrumental battle pavan (b).

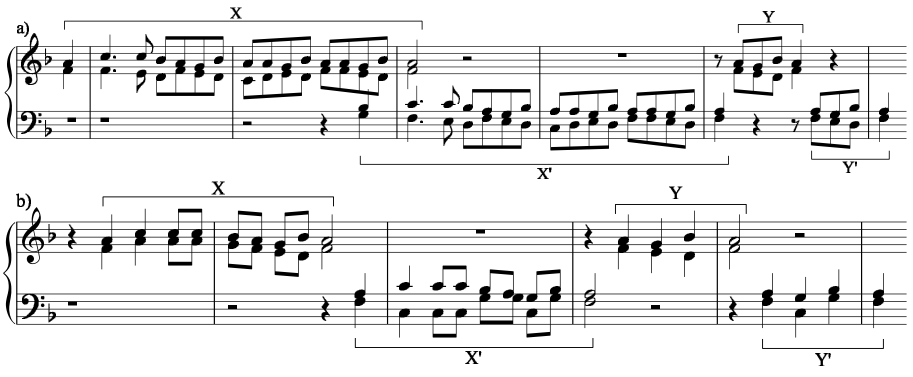


Fig. 2. Extracts from (a) Janequin, *La Guerre* ([17], pp. 24-25, text removed here for clarity) and (b) Moderne, *Pavane. La Bataille*, showing the same 'antiphonal' feature ([18], p. 23).

This is a compound feature, which we shall call the 'antiphonal' feature: a phrase (X), which is subsequently imitated an octave (sometimes two octaves) below (X′), then a shorter phrase derived from the tail end of its predecessor (Y) and also similarly echoed (Y′). In Janequin's original, only the first chord of the X′ imitation is changed, but Moderne radically alters the lower voice. Although X may vary between pieces, there is enough common ground to construct a melodic outline which can form the basis of a query. Despite the varied amount of time separating each phrase from its echo, the duration elapsing between X and X′ is always greater than that between Y and Y′.

So, to abstract this verbal description – to make it modular and each modular element reusable – we identify two descriptive elements: `Melody`, which describes pitch and rhythm relationships; and `Imitation`, which describes the application of a process of transformation to a phrase. These elements are combined either simultaneously or in sequence, i.e. they are either `Concurrent` or `Consecutive`.

We can represent this particular feature-description in a block diagram (Figure 3). This diagram is a graphical abstraction of the 'antiphonal' feature, which can (when specified more formally) now be added to a list of features which are characteristic of battle pieces and may be used in identifying them. As further features are added together with descriptions of their relationships, we begin to accumulate an ontological vocabulary for describing a set of pieces against the background of a broader repertory. As additional studies are added, the vocabulary of this ontology gains general descriptive power.

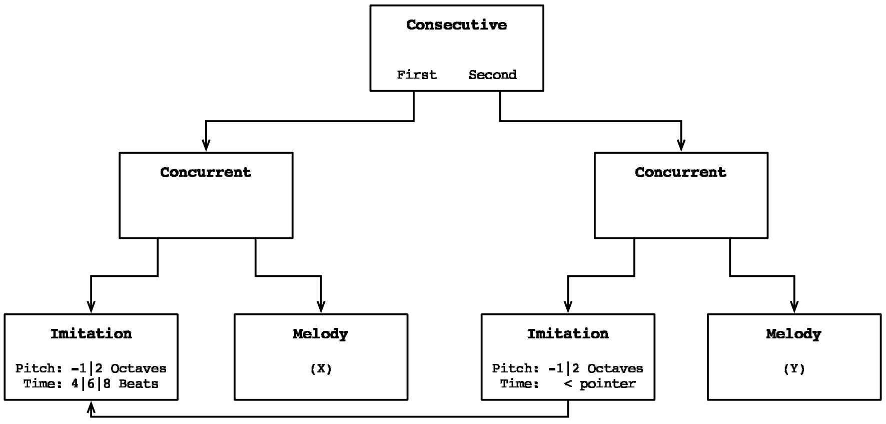


Fig. 3. Block diagram illustrating the combination of features to form a description of the 'antiphonal feature'. Arrows indicate pointers within the structure, either to other modules (as in the case of the combining elements `Concurrent` and `Consecutive`) or to values within other modules (as in the case of the second `Imitation`, which compares its parameters with those of the first).

4 Corpus Interaction

Many different MIR algorithms and tools are now in use and/or described in print and, although all the tools used in the 'battles' project were written from scratch, several could in principle have been derived from pre-existing work (had that work been appropriately implemented). Potentially, much of the task of enabling interaction with a corpus consists of mapping existing MIR tools on to the ontology outlined above – one can consider the ontology as a means of abstracting human descriptions of music and the MIR tools as a way of doing the same for machine representations of music. The issue, then, is mapping between the two abstractions.

In both cases, there is an element of interpretation and a real danger of data loss. The features descriptions, in their abstracted form, in aiming for maximum generalizing power are likely to have lost some of the subtlety of a plain verbal form, whilst the choice of applicable MIR tools is also bound to be an interpretative process. Musical terms rarely have concrete definitions, nor are the terms necessarily fixed with respect to time or context; in order to use them, it is necessary to come to some general agreement about their use in certain contexts.

Different corpora will require different tools or different configurations and combinations of those tools. Some research questions are self-limiting, being only applicable to certain musics or domains: a survey of pitch spelling as pieces move from flat to sharp keys, for example, can be conducted in the symbolic domain but not in the audio domain, since a recorded pitch (frequency) rarely allows one to distinguish, say, a C♯ or a D♭; on the other hand, a study of intonation – whether a performer plays sharp or flat of some reference pitch – would only be applicable to audio. A reference to the use of a certain fingering, however, might not be so straightforward: in symbolic representations, it might be explicit – marked in – or it might be implicit – if an earlier note, or group of notes, has a fingering-pattern marked that suggests it might be applicable elsewhere – or it might be deduced by some algorithm (see, for example, [19]); in audio, such information might conceivably be deduced from timbral information from recordings of some instruments – but it will not be explicit.

Not all these differences are to do with the medium or domain of transmission, however. For example, in the battle music we are dealing with, notated durations are frequently scaled by a factor: the second phrase in the antiphonal feature (Figure 1), for example, is written in quavers in Janequin's original but in crotchets in the Moderne version, although the pitches of the melody are identical.[5] A melodic feature search appropriate for this corpus, then, also needs to work for augmentations (expansions by a factor) and diminutions (reductions by a factor) of the query's rhythm.

5 Generalised Nature of Features

Much of the power offered by our proposed approach arises from the extent to which features may appear in many different contexts, retaining some sense of identity even where their role in the music is substantially changed.

This may be illustrated by reference to another common feature in battle pieces, the musical depiction of a fife (small flute) and drum. Figure 4 shows an extract from William Byrd's keyboard piece, *The Battell*, which has all of its main features, and is even explicitly captioned 'The Flute and the Droome'.

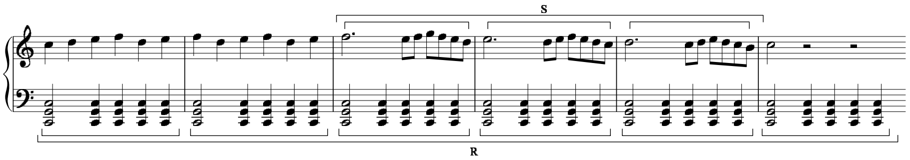


Fig. 4. Extract from Byrd's *The Battell* (from *My Ladye Nevell's Booke*, [20], p. 30), section entitled 'The flute and the droome'

In this extract, the sound of the drum is represented by repeating a rhythmic pattern (labelled R) on an unchanging low chord, a drone consisting of the keynote with a fifth and octave added above it. The fife is represented as a high-pitched melody, moving quickly in relation to the general pulse of the music. The fife melody varies from piece to piece, but tends to move almost entirely by step or small skip and to end with a descending sequence (labelled S).

A block diagram showing how this feature description might be expressed is given in Figure 5. Any of the constituents of this feature, taken separately, could

[5] These are originally crotchets and minims respectively in the sources, but note durations have been halved in the examples given for ease of reading.

certainly not be used to locate it, and merely identifying each of them within a single piece would also be insufficient; we need to be able to specify further that they occur simultaneously, or at least in close proximity – hence the use of the combining element `Concurrent` in Figure 5. Taken individually, each constituent feature can be associated with a host of other uses and meanings, often depending heavily on context, and thus could form part of many varied queries. Repeating rhythmic patterns form the basis of many dance forms – most famously, perhaps, the snare-drum pattern running continuously throughout Ravel's *Bolero*. Other queries might incorporate searches for a repeating rhythm, unspecified (as here) or specified (as in a search for a particular dance).

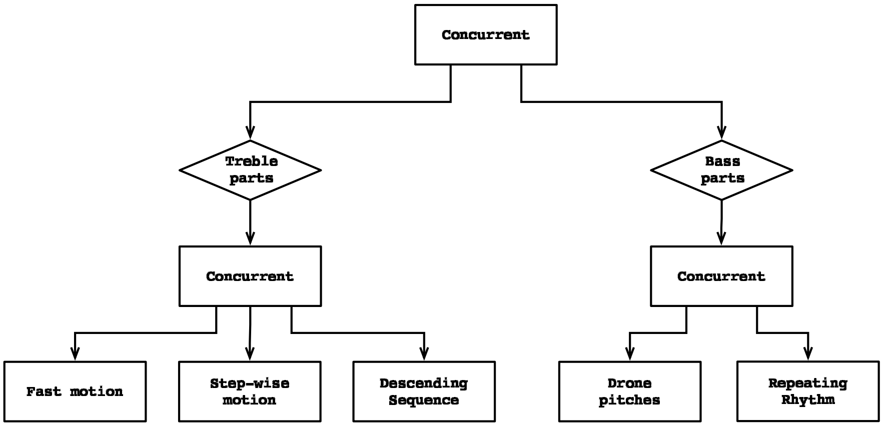


Fig. 5. Block diagram illustrating the combination of constituent features to form a description of the 'fife and drum' compound feature. Rhombus-shaped blocks are filters, in this case filtering for the 'treble' and 'bass' pitches or voices.

Similarly, the use of a drone separated from the 'drum' rhythm is often seen as a marker for 'rustic' music, conjuring up images of bagpipes and hurdy-gurdies. The 'pastoral music' topos, most famously used in Beethoven's Sixth Symphony, makes extensive use of the drone, along with many other features that could be used for retrieval of pastoral music, including imitations of birdsong and other animal noises, the use of compound meter, and many other localized rhythmic and melodic markers. Thus, although the larger-scale feature descriptions may be unique to a particular expression of a query, the individual units would be expected to receive significant reuse, and this reuse could easily occur in the context of very different music.

6 Conclusion and Future Directions

We have suggested throughout this paper that a user need not be (indeed, most users *should not* be) routinely exposed to the details of the internal data representations used in corpora to perform information retrieval tasks. By employing a layer of abstraction between the user and the corpus-specific search tools, we can create a representation of a query that is (largely) independent of the format of any given resource. If a sufficiently stable, standardised core vocabulary can be established – by accumulating elements from many musical communities – then it is to this that developers of both user interfaces and music resource search tools could look, thus allowing a substantial degree of interoperability.

Thus far, no mention has been made of one of the most important elements in music descriptions: textual metadata. Almost every musicological query, whether expressed on a computer or elsewhere, will contain components that arise not from the music itself, but from extrinsic information such as title and attribution information, dates and details of provenance, performer's names or references to concordances. All this – along with information derived from the notated music, analytical information from research, and any other relevant data – we refer to under the blanket term *metadata*. Such data forms a vital foundation for supporting further investigation but can be very complex. As we have seen to be the case for musical data, there are many ways of describing the same thing and a single description can have many possible meanings; there are also many standards for representing the data itself and widely divergent requirements arise from different communities.

The mechanisms and structures for metadata in the abstraction layer should be the same as (or at least compatible with) those for the vocabulary based on the musical data. In many cases, there is likely to be some overlap: with a piece's key, for example, being extrinsic metadata in one resource but derived from the music content in another. In this case, the vocabulary term 'key' needs to be able to accommodate both origins transparently.

With more research on the nature and variety of human musical description and musicological practices, we can expect to gain a better understanding of the ontology and structures that a system such as we describe here will require. Once these structures are well enough understood for us to create a general specification, it should be possible to build a system sufficiently powerful to demonstrate the principles of generalization and interoperability we describe here.

References

1. Gruber, T.: What is an Ontology? `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html` (1998)
2. Lewin, D.: Generalized Musical Intervals and Transformations. Yale University Press, New Haven and London (1987)
3. Wiggins, G., Harris, M., Smaill, A.: Representing Music for Analysis and Composition. `http://www.doc.gold.ac.uk/~mas02gw/papers/EWAIM89.pdf` (1989)
4. Conklin, D.: Representation and discovery of vertical patterns in music. In Anagnostopoulou, C., Ferrand, M., Smaill, A., eds.: Music and Artificial Intelligence: Lecture Notes in Artificial Intelligence. Volume 2445. Springer Verlag (2002) 32–42

5. Smaill, A., Wiggins, G., Harris, M.: Hierarchical Music Representation for Composition and Analysis. Journal of Computing and the Humanities **27** (1993) 7–17
6. Michael Kassler: Toward Music Information Retrieval. Perspectives of New Music **4** (1966) 59–67
7. Page, S.D.: Computer Tools for Music Information Retrieval. PhD thesis, New College and Programming Research Group, University of Oxford (1988)
8. Huron, D.: What is a Musical Feature? Forte's Analysis of Brahms's Opus 51, No. 1, Revisited. Music Theory Online **7** (2001)
9. Pople, A.: Modelling Musical Structure. In Cook, N., Clarke, E., eds.: Empirical Musicology. Oxford University Press, Oxford (2004) 127–156
10. Kuhn, T.: On the Structure of Scientific Revolutions. University of Chicago Press (first edition 1962, postscript added for second edition 1969)
11. Bainbridge, D., Cunningham, S.J., Downie, J.S.: How People Describe their Music Information Needs: A Grounded Theory Analysis of Music Queries. In: ISMIR 2003: 4th International Conference on Music Information Retrieval: Proceedings, Baltimore (2003) 221–222
12. Lee, J.H., Downie, J.S.: Survey of Music Information Needs, Uses and Seeking Behaviours: Preliminary Findings. In: ISMIR 2004: 5th International Conference on Music Information Retrieval: Proceedings, Barcelona (2004) 441–446
13. Gale, M.: What's in a Battle?: Identifying a popular Renaissance Genre. Presented at the 40th Annual Conference of the Royal Musical Association (2004)
14. Brown, A.: Battle Music. In Sadie, S., Tyrrell, J., eds.: The New Grove Dictionary of Music and Musicians. Volume 2. Macmillan, London (2001) 915–917
15. Gale, M., Lewis, D.: "La battaglia": a computer-assisted approach to an extended musical family. Presented at the 51st Annual Conference of the Renaissance Society of America (2005)
16. Crawford, T.: Applications Involving Tablatures: *TabCode* for Lute Repertories. Computing in Musicology **7** (1991) 57–59
17. Clément Janequin: La Guerre. In Merrit, A.T., Lesure, F., eds.: Clément Janequin: Chansons polyphoniques. Volume I. Oiseau Lyre, Monaco (1965)
18. Jacques Moderne: Pavane. La Bataille. In Giesbert, F.J., ed.: Musique de joye = Fröliche Musik: eine Folge alter Tanzstücke... Nagels Verlag, Kassel (1970)
19. Parncutt, R., Sloboda, J.A., Clarke, E.F., Raekallio, M., Desain, P.: An Ergonomic Model of Keyboard Fingering for Melodic Fragments. Music Perception (1997) 341–381
20. William Byrd: The Battell. In Andrews, H., ed.: My Ladye Nevells Booke of Virginal Music. Curwen & Sons, London; repr. Dover Editions, New York (1926; 1969)

An Editor for Lute Tablature

Christophe Rhodes and David Lewis

Centre for Cognition, Computation and Culture,
Goldsmiths College, University of London,
New Cross Gate, London SE14 6NW, UK
c.rhodes@gold.ac.uk, d.lewis@gold.ac.uk

Abstract. We describe a system for the entry and editing of music in lute tablature. The editor provides instant visual and MIDI feedback, mouse and keyboard controls, a macro recording facility, and full run-time extensibility. We conclude by discussing planned future functionality and considering other potential applications for the technology.

1 Introduction

The Electronic Corpus of Lute Music[1] seeks to act as a first point of reference for researchers in lute music and to raise the profile of the repertoire. The Western European lute is an instrument of great historical importance, and estimates [1] suggest a still extant repertory of nearly 60,000 pieces scored for the instrument. Yet, despite its clear significance, the lute and its music play a comparatively minor part in current musicology: the music is not generally well-known and its historical rôle rarely discussed.

This obscurity arises in part because the surviving sources are often miscellanies, making location of works by a single composer or finding sources for a specific piece difficult. More significant, perhaps, is the fact that the notations for lute music are very different from staff notation: anyone curious to explore the repertory must first learn the notation and, until they become experienced, to transcribe the pieces into a more familiar format.

ECOLM [2] aims to make it easier for lute music to be appreciated by non-experts, without the need to understand fully either tablature or the technique of the instrument, whilst still providing scholars with the detailed, specialist information where it is required (see [3] for an example). The core of the project consists of musical encodings, forming an online edition, providing 'diplomatic facsimiles' (*i.e.* literal transcriptions), editions, and computer-generated MIDI and staff transcriptions. We also seek to create an infrastructure for distributed editing, necessitating an encoding system and a user interface.

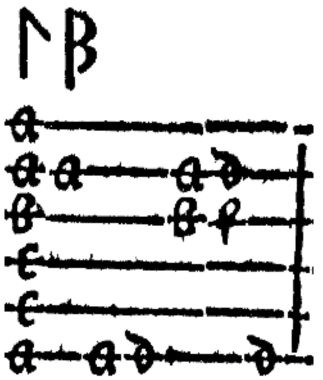
2 Lute Tablature and *TabCode*

Lute music is written in a variety of different tablature forms. The notation tells the lutenist which strings should be played at which frets and when. The most

[1] ECOLM 2 is the second phase of a five-year grant provided by the UK Arts and Humanities Research Board (now Council).

common approach is to use horizontal lines (similar to staff lines) to represent strings, and letters or numbers placed on them to represent frets – a practice that survives to this day in guitar tablature. The music is read left to right, with rhythm signs placed above to indicate timing. At its most basic, lute tablature is entirely sequential in nature, with one chord following another without notated overlap, and it is impossible to indicate multiple simultaneous rhythms.

Clearly, this attribute makes the transfer of the notational information into ASCII for use in the corpus a much simpler proposition than has been the case for the majority of classical music scores. Crawford [4] describes a format called *TabCode* for encoding lute tablature as a series of 'tabwords' separated by white space. Each word begins with a character indicating the rhythm sign (the initial of the name of that sign: `H` for half note, `Q` for quarter, etc.), if present, followed by a letter-number pair for each symbol in the chord, with the number signifying the string and the letter the fret.



```
Qa1a2b3c4c5a6
Ea2
a6
d6
a2b3
d2f3
d6
|
```

Fig. 1. An extract from 'Fantasia Ioannis Dooland Angli *Lachrimae*', Jean-Baptiste Besard, *Thesaurus Harmonicus* (1603), f.16v, and its *TabCode* encoding

TabCode was designed as a terse, human-readable language, enabling fast input and editing whilst still preserving descriptive power. Encodings of lute music in other tablature representation languages such as `abctab` [5] and `**fret` [6] can be converted trivially to *TabCode*, which contains a richer vocabulary for this repertory. For a more sophisticated set of text-critical and editorial tools, we are working with Frans Wiering to produce a mark-up for *TabCode*, TabXML, drawing from TEI[7] and MEI[8] standards[2].

In order to create camera-ready artwork, in particular for [10], Tim Crawford wrote The Tablature Processor, a Macintosh-based application with its own binary file format, but capable of importing and exporting *TabCode.* The Tab Processor has some facility for data entry, but is primarily a type-setting program for a type of lute tablature.

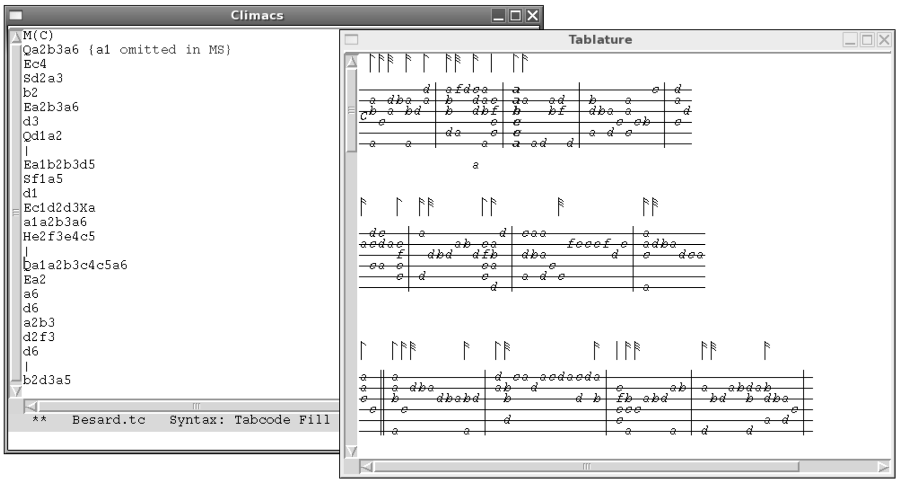
[2] For a discussion of TabXML and its relationships with the standards cited, see [9].

3 TabEditor

We have developed a new application to speed the process of entering and editing *TabCode*. This application provides a syntax-aware editor in the Emacs tradition, along with a graphical view of the currently-edited score, which updates in real-time as the user manipulates the textual *TabCode* buffer. In addition, this graphical view is mouse-sensitive (as is the editor buffer) and allows interaction with the graphical objects, while retaining the primacy of the text representation: manipulation of the graphical object is implemented as a sequence of manipulations on the text. We also allow the option for the user to receive immediate audio feedback of the current chord at its completion, as well as region or full-piece audio rendering.

The major improvement in this application compared with previous editors is the immediacy of the feedback; however, from the editor's Emacs heritage comes extensibility, both through explicit definition of additional functionality at run-time and through the ability to record keyboard macros for automation of repetitive tasks.

This application is not at present able to produce an edition-ready rendering of *TabCode*, unlike the Tablature Processor; this stems directly from the fact that the textual *TabCode* is required to contain all the information in the application: the language is rich enough to express the semantics of a lute tablature manuscript, but not the necessary tweaks that an editor makes for the print copy of a work. However, it is capable of rendering the *TabCode* to draft-quality PostScript or to an image for further processing, display or printing, and is used on-demand for this purpose in the ECOLM Web frontend.



```
M(C)
Qa2b3a6 {a1 omitted in MS}
Ec4
Sd2a3
b2
Ea2b3a6
d3
Qd1a2
|
Ea1b2b3d5
Sf1a5
d1
Ec1d2d3Xa
a1a2b3a6
He2f3e4c5
|
Qa1a2b3c4c5a6
Ea2
a6
d6
a2b3
d2f3
d6
|
b2d3a5
```

```
**   Besard.tc   Syntax: Tabcode Fill
```

Fig. 2. The tablature editor and graphical display. Note the third bar of the tablature, which corresponds to the fragment in figure 1.

3.1 Implementation Details

The application is written in Common Lisp, using the SBCL [11] environment, the McCLIM [12] implementation of the CLIM specification for graphical interface management, and Climacs, a CLIM editor, as our editor substrate. It is beyond the scope of this paper to give a detailed exposition of CLIM's capabilities: we refer the interested reader to other sources such as [13]. For our purposes, CLIM associates graphical output with application data through *presentations*, and manages the efficient redrawing of application state through *incremental redisplay*.

The CLIM presentation facility provides what is in some sense an object-oriented renderer: when drawing the graphical view of the tablature denoted by the *TabCode*, the graphical elements retain their association with the `tabword` objects that result from parsing the editor buffer. This then allows a trivial implementation of actions such as moving the cursor to the point in the buffer corresponding to a particular chord in the graphical view, and a relatively simple implementation of commands for musical manipulation (such as one to move glyphs up a string to correct a typographical error) operating on the textual *TabCode* but triggered by an action in the graphical view.

Although the Climacs editor includes a syntax analysis module [14] based on a parser implementing Earley's algorithm [15], the generality of this framework was not needed for an encoding language as simple as *TabCode* (which is fundamentally a regular language). Instead, we implemented a combined lexer and parser, which on a parse error preserves the partial parse, if any; advances to the nearest lexically following whitespace; and resets the analyser's state.

This permissive behaviour on a *TabCode* parse error is motivated by the observation that textual editing will tend to cause the buffer to go through locally invalid states during the course of one logical edit (*e.g.* entering a single tabword), and that it would be distracting to the user to cause this local invalidity to propagate to the global state of the parser as displayed in the rendered tablature window.

This parser generates a sequence of `tabwords` from the text in the editor buffer, reusing portions of the previously-generated sequence (active before the edit) if it can prove, based on the extent of the text region 'damaged' by user interaction, that the parse is unchanged.

Incremental redisplay is in some sense merely an optimization, but it is a sufficiently broad one that it merits discussion: it permits the system to avoid redrawing output if it can determine that this will not be necessary, on an object-by-object basis. We use this optimization by preserving the identity of those elements of the parsed buffer contents which can be proved not to have changed, as described above. This cache not only informs the display within the editor buffer – highlighting parse errors, for example – but also the graphical view: a chord need not be redrawn if it has not changed since the last edit.[3]

[3] In addition to this, it should also not have changed its position. This is the case for many edits in a *TabCode* document: only those which change an element's width will affect the positioning of subsequent elements on that line.

4 Conclusions and Future Directions

The real-time feedback provided by this application has met with approval from its users, including some with limited technical skills: errors are corrected more quickly, and the ability to find the area in the *TabCode* source corresponding to a place in the graphical output allows more efficient navigation. The CLIM framework assisted us in development of this application by providing the means to associate high-level application data directly with the graphical output.

TabCode is a simple language when interpreted as a sequence of tabwords representing notated elements. However, for general semantic analysis and display purposes, a slightly more sophisticated parsing framework is required: to accommodate hierarchical groupings such as beaming, connecting lines and so on tabword groups must be formed. These hierarchical groups can be incrementally maintained in the same manner as the sequence of tabwords, by computing the overlap of groups with the 'damaged' region in the editor buffer.

A feature planned for the near future is transcription of the current buffer to score (in Common Music Notation). The initial transcription algorithm should prove fairly simple, as each chord can simply be directly transcribed; difficulties remain in the areas of pitch spelling and readable, musically sensible, polyphony.

We believe it would be relatively simple to adapt our application to support other textual representations of music (such as Humdrum's [6] `**kern`) or more generally of two-dimensional data, while maintaining the association between graphical display and textual input. Such an adaptation would need to provide only a resynchronising parser and a renderer for parse trees.

Acknowledgments

C.R. and D.L are supported by EPSRC grant GR/S84750/01 and AHRC grant B/RE/AN9717/APN15559 respectively.

References

1. Ness, A.J., Kolczynski, C.A.: Sources of lute music. In Sadie, S., Tyrrell, J., eds.: The New Grove Dictionary of Music and Musicians. Volume 23. Macmillan, London (2001) 39–63
2. Crawford, T., Gale, M., Lewis, D.: An Electronic Corpus of Lute Music (ECOLM): technological challenges and musicological possibilities. In Parncutt, R., ed.: Conference on Interdisciplinary Musicology, Graz (2004) 118–119
3. Lewis, D., Gale, M.: "La battaglia": a computer-assisted approach to an extended musical family. Presented at the Annual Conference of the Renaissance Society of America (2005)
4. Crawford, T.: Applications Involving Tablatures: *TabCode* for Lute Repertories. Computing in Musicology **7** (1991) 57–59
5. Dalitz, C.: abctab2ps Users' Guide. `http://www.lautengesellschaft.de/cdmm/userguide/userguide.html` (2005)
6. Huron, D.B.: The Humdrum Toolkit. CCRAH, California. (1994)

7. The Text Encoding Initiative Consortium: TEI: Yesterday's Information Tomorrow. `http://www.tei-c.org/` (2005)
8. University of Virginia: The Music Encoding Initiative (MEI). `http://www.lib.virginia.edu/digital/resndev/mei/` (2004)
9. Wiering, F., Crawford, T., Lewis, D.: Creating an XML Vocabulary for Encoding Lute Music. In: Proceedings of the XVIth International Conference of the Association for History and Computing, Amsterdam (forthcoming, 2005)
10. Crawford, T., ed.: Silvius Leopold Weiss: Sämtliche Werke für Laute. Volume 5–7. Bärenreiter, Kassel (2002–)
11. Newman, W.H., et al.: SBCL User Manual. `http://www.sbcl.org/manual/` (2000)
12. Strandh, R., Moore, T.: A Free Implementation of CLIM. In: International Lisp Conference, San Francisco, Franz Inc. (2002)
13. Rao, R., York, W.M., Doughty, D.: A guided tour of the Common Lisp interface manager. ACM SIGPLAN Lisp Pointers **4** (1990) 17–37
14. Rhodes, C., Strandh, R., Mastenbrook, B.: Syntax Analysis in the Climacs Text Editor. In: International Lisp Conference, Stanford (2005)
15. Earley, J.: An Efficient Context-Free Parsing Algorithm. Communications of the ACM **13** (1970) 94–102

Interdisciplinarity and Computer Music Modeling and Information Retrieval: When Will the Humanities Get into the Act?

Cynthia M. Grund

Institute of Philosophy, Education and the Study of Religions Philosophy,
University of Southern Denmark,
Odense, Denmark
cmgrund@ifpr.sdu.dk

Abstract. This paper takes a look at computer music modeling and information retrieval (CMMIR) from the point of view of the humanities with emphasis upon areas relevant to the philosophy of music. The desire for more interdisciplinary research involving CMMIR and the humanities is expressed and some specific positive experiences are cited which have given this author reason to believe that such cooperation is beneficial for both sides. A short list of some contemporary areas of interest in the philosophy of music is provided, and it is suggested that these could be interesting areas for interdisciplinary work involving CMMIR. The paper concludes with some remarks proffered during a panel discussion which took place near the end of the Pisa conference on September 28, 2006 and in correspondence inspired by this discussion, together with some brief commentary on the same. An earlier, somewhat short version of the present paper provided the impetus for said panel discussion.

1 Introduction

Traditionally, the extent to which the *substantial* content in humanistic discourse about music – such as that produced within musicology, philosophy, aesthetics, and vocal and instrumental pedagogy – has been influenced by the *thinking* behind developments in electronic technologies for sound reproduction, storage and production has been very limited, if not negligible. The dominant attitude on the part of the humanities regarding the relationship of technology to theorizing about music and to musicianship has been one of handmaidenship: technology enabled us to hear performances again, to have access to performances we otherwise would not have had a chance to hear, and it could help to compensate for inadequate acoustics in concert venues by providing us with amplification. All the intellectually respectable and culturally important cogitating from the point of view of the humanities, however, was that which dealt with the questions of the sonic content of these recordings and performances; the recordings and amplified performance were, if anything, inferior to live, "acoustic" ones (the appropriate modifications of this remark applying *mutatis mutandis* to jazz, rock, etc.) and the theorizing which was involved in developing and achieving the results afforded by the various technologies was seen as the purview of engineers, acousticians, and other applied-science types.

The foregoing remarks are intended to set the tone for some investigatory musings regarding the ways in which computer music modeling and [information] retrieval (CMMIR) may be seen as a field of technological development which has the potential for changing this relationship between humanistic and technological approaches to music. Since I am situated within the humanities as a member of a philosophy department, my thoughts will mainly have the philosophy of music and related areas of aesthetics as their point of departure. These deliberations are what will constitute the principle theme of this paper, and they are the matters to which we will now turn.

2 Background

The alert reader is perhaps already asking him- or herself how the author of this paper, herself a philosopher, ever became sufficiently aware of what was going on the CMMIR side of the fence to regard the research being done there as relevant to humanistic research. Since this is a good question, I will take the time to answer it here.

My ongoing fascination with computer music modeling and [information] retrieval can probably be dated back to some presentations which were given by fellow participants – particularly Thomas Noll and Marc Leman – at *Music and Logic*, The Twelfth Meeting of the FWO Research Society on Foundations of Music Research at the Institute for Psychoacoustics and Electronic Music at Ghent University at the beginning of 2001. [1] Later on in 2001 the Danish Research Council for the Humanities provided two year's worth of funding to start up the research network NTSMB, *Netværk for Tværvidenskabelige Studier af Musik og Betyding/Network for Cross-Disciplinary Studies of Music and Meaning.* [2] CMMIR made its debut within NTSMB during the international conference *Nature, Culture and Musical Meaning* in the summer of 2002 with Marc Leman's presentation "Musical Meaning Formation from the Viewpoint of Microscopic Musicology." It played a major role in the third national meeting of NTSMB – *Musik, Logik og Teknologi/Music, Logic and Technology in* November 2002, where the presentations given included papers by Anders Christian Gade and Christoffer A. Wietze, Esben Skovenborg, Kjetil Falkenberg Hansen, Tony Brooks and Declan Thomas Murphy.[3] It is relevant to mention that this is one of the most successful and exciting meetings we have held under the aegis of NTSMB, in spite of the fact that quite a few of the network members from the humanities had expressed reservations about a theme such as this one, not in the least due to their fear that they wouldn't be able to understand anything. Nothing could have been further from the truth. People in the audience from the humanities told me afterwards that

[1] See http://www.ipem.ugent.be/research/nfwo/

[2] See http://www.ntsmb.dk for programs of previously held conferences and announcements of upcoming conferences.

[3] See http://www.ntsmb.dk

they had been given food for thought to a degree they had not believed possible, and representatives from the technical side of the fence revealed that they found speaking before a group like this to be a liberating and exciting experience. Although we have not had such a concentration of technical talks during the ensuing biannual meetings, there have been technical talks sprinkled throughout the programs and people with technical backgrounds in the audience, largely due to the success of the November 2002 meeting. That technical and traditionally humanistic approaches can complement and supplement each other has become evident within the context of the peer-reviewed online journal, *JMM: The Journal of Music and Meaning*, which is an outgrowth of NTSMB.[4]

One of the reasons that this joining of the humanities and technology took place was that the emergence of NTSMB on the Danish scene attracted the attention of Jens Arnspang of Aalborg University at Esbjerg, who suggested that we establish some cooperation with the MOSART (Music Orchestration Systems in Algorithmic Research and Technology) network. Ultimately, common interests led to an application to The Danish National Research Foundation for five years of funding for a Center of Excellence. The suggested name for the center was "Research Center for Music, Modeling and Meaning." It was envisioned as a meeting place for humanists and for experts in CMMIR to meet and cooperate in research within (1) linguistic representation of sound, (2) sound, music and bodily gesture, (3) recognition and interpretation of form in sonic contexts, (4) issues in composing, (5) the creation of meaning in interactive sonic contexts, (6) the role of sound in creating a meaningful environment for human agents, (7) perception of sonic and visual phenomena, and (8) practice-based research: problems and perspectives. Our proposed center was among the 177 which were not funded. 16 centers were; only one of them had any appreciable connection to the humanities. So, the rejection of this application was par for this course, and any discouragement on that score has to be adjusted accordingly. What I found to be manifestly encouraging, however, were the exciting ways in which problems of the humanities and work within CMMIR dovetailed.

Why aren't there more projects of this sort? Of course, the first answer which immediately suggests itself is already provided in the foregoing: lack of funding. So let's roll back one step and ask: Why is it hard to persuade those with their hands on the purse strings that CMMIR is a place where the humanities and technology can meet and exciting things can happen as a result?

3 Interdisciplinarity and CMMIR

3.1 Obstacles?

Some of the obstacles to cooperative projects involving the humanities and CMMIR are self-imposed and can be removed from among our own ranks on both sides. The burden of removing barriers to cooperation must be shared by both sides, however. It

[4] See http://www.musicandmeaning.net

is, for example, interesting to note the nature of the interdisciplinarity sketched on the homepage for this conference:[5]

> **"BACKGROUND**
> The field of computer music is interdisciplinary by nature and closely related to a number of computer science and engineering areas such as information retrieval, programming, human computer interaction, digital libraries, hypermedia, artificial intelligence, acoustics, signal processing, etc...
>
> In this year's CMMIR we would like to emphasize on the human interaction in music, simply the PLAY, meaning that papers related to sound modeling, real-time interaction, interactive music, perception and cognition are particularly welcome together with the usual themes related to the traditional topics of the CMMR conference."

It is as if CMMIR as an area of research is unaware of its own potential for reaching out or appealing to areas within the more traditional areas of humanistic music research. Of course, one might harbor the prejudice that people educated and trained within the sorts of sophisticated techniques required for solving the technical problems encountered in CMMIR would be *a fortiori* the only ones with anything to contribute to the formulation of problems which the field might want to address. I must admit that it is easy to fall prey to this prejudice given the massive disregard there has been – and often still is – for formal and mathematical training within certain sectors of the humanities. The attitude is - although unfortunately justified in some quarters - unduly negative. As indicated in the foregoing, my experiences with various meetings within NTSMB, the response to JMM and the great enthusiasm and effort put forth by all involved in the Center-of-Excellence application, suggest that the areas which are ripe for productive cross-fertilization are many.

The burden of proof will still lie with those within the traditional areas of CMMIR research and those of us from the humanities who have discovered what CMMIR is capable of to work together both in doing research and in formulating projects. This must be done in such a way that even those within the CMMIR community who are skeptical of what humanists can contribute and those within the humanities with limited formal skills can see the interesting prospects for cooperation. It is far from given that those who sit in positions of power in funding and granting agencies are to be found among the intersecting group of researchers within CMMIR and within the traditional humanities who have seen the light. Many will undoubtedly see CMMIR through the lens of hard science and evaluate it positively to the extent that it can produce marketable technologies. Those who see it through the lens of the humanities may very well also evaluate it as a source of marketable technology, see this as something negative, and go no further, since "success" in the humanities has – at least on the face of it - much more to do with the winning over of minds and hearts (and number of pages published/books sold) than in the production of marketable technology.

[5] See http://www.lma.cnrs-mrs.fr/~cmmr2005/

3.2 Reflections upon – and Suggestions for – Ways in Which Computer Music Modeling and Retrieval Could Recontextualize (and Rejuvenate) the Philosophy of Music (with Some Positive Influence Running in the Other Direction as Well…)

Since I come from philosophy, I will stick with philosophy of music as a source for examples of where the interests cultivated within CMMIR and interests cultivated within traditional humanistic music research could dovetail:

1. Philosophy is still casting about for an understanding of language which can account for its dynamic, interactive online nature. Granted, moves have been made to augment the view of language as *essentially* an abstract mathematical structure:
   - 1.1. More attention is – finally – being paid to its development in human beings who are situated in a sounding world for whom language is for many years of early development a sensuous matter of the re-establishment of intimacy with a mother to whom one is an external rather than internal being and the establishment of intimacy and social bonds with a whole cast of other humans. The work being done to study the employment of "motherese" between a baby and its mother suggests fascinating prosodic resemblances between our earliest experience of what could be termed a proto-language and many established and preferred organizational schemes for note and phase partitioning within music across cultures. [6]
   - 1.2. Speculative research is also in progress to try to account for how we as a species ever managed to develop language in the first place; prosodic and subsequent rhetorical features of language based upon sonic qualities are vying for a place in the pack in which, at least where philosophers were concerned, matters of reference and denotation where the lead dogs. [7]
   - 1.3. There is also extremely exciting research being done to see if signify-cant correlations can be established between sonic properties of a language group and the compositional music which its representatives have produced.[8]
2. The role and significance of memory in the development of human cognition, culture and behavior has finally been recognized.[9] Since music clearly has a place in, among other areas of human endeavor, the repertoire of mnemonic devices and strategies we have developed throughout millennia, study of "memorability" in terms of patterns and the like is a clear area for CMMIR-inspired research.
3. As human corporality becomes more important as a factor in philosophical considerations involving the genesis of meaning and intentionality, the role of gesture in human interaction with the environment also becomes an increasingly

[6] See the – extremely selective list of – references for Malloch and Trevarthen.

[7] See Christensen-Dalsgaard.

[8] See Magne et al. for up-to-the-minute CMMIR work in this area.

[9] For a glance at what is just the tip of the iceberg, fascinating insights into the way in which the need to remember can account for religious behavior for which it is otherwise very diffi-cult to give an account, are provided in Whitehouse, 2000.

relevant area of study for philosophers. Work which addresses this area has already been done within CMMIR research.[10]

4. There are also very down-to-earth ways in which CMMIR could have a salutary influence on research within the philosophy of music: as researchers socialized within the behavioral paradigms of the way research is done in the natural sciences, people working within traditional CMMIR are used to working in groups and solving problems as a matter of teamwork. Within the traditional humanities – and philosophy is a paradigm case – there is still an ideal for research which involves the sovereign, solo thinker working in splendid, eremitic isolation. Although this very well may be a modus operandi which is suitable for some limited class of problems, given the complexity of contemporary knowledge and the at times overwhelming amounts of it that should be taken into consideration when doing high-level research in an area involving something as multifaceted as music, it would serve many humanists well to take some cues from the established CMMIR community.

4 Reality Check

This contribution to these proceedings is the result of having reflected upon the research within CMMIR with which I have been familiarizing myself during the past several years through reading, attending conferences such as CMMR2004 in Esbjerg, Denmark in May 2004 and ISMIR2004 in Barcelona, Spain in October 2004, and through my role as director of NTSMB and editor-in-chief of *JMM: The Journal of Music and Meaning*. In the course of both formal and informal colloquia and discussions, it has time and time again been evident to me that the CMMIR community is yearning for more feedback from the humanities community, and that the humanities community can derive tremendous benefit from heightened exposure to and understanding of what is going on in CMMIR.

It was my hope that these few pages would contribute to inspiring some sort of dialog between the two research communities. As it happened, a panel discussion which I had the pleasure of moderating at the conclusion of CMMR 2005 initiated some discussion of this sort. The participants were Tim Crawford of Goldsmiths College, University of London; Laurent Daudet, LAN, U. Pierre et Marie Curie (Paris 6), Paris, France; Kristoffer Jensen, Aalborg University Esbjerg, Esbjerg, Denmark; Leonello Tarabella, CNR, Pisa, Italy and Sølvi Ystad, CNRS-LMA, Marseille, France. Audience participation was enthusiastic, and I received some e-mails afterwards as well, both from members of the panel and from audience participants. Several themes emerged throughout the discussion and ensuing correspondence. Here is an overview.

Daudet made some remarks before and after the panel discussion which are highly relevant, since they honestly reflect the pros and cons attendant to interdisciplinary work. In an e-mail following the panel discussion, he noted that although much lip service is paid to the promotion of interdisciplinarity within universities, more often than not anything that crosses departmental lines runs into funding difficulties. Just as sobering is the fact that if results do come of this sort of research, they are not

[10] See, for example, Murphy and Tarabella.

acknowledged as "really serious research." His personal estimate was that three articles in a publication such as *JMM: The Journal of Music and Meaning* would have less value than one in the *IEEE Transactions on Signal Processing* for someone such as himself – a researcher working in an acoustics laboratory – with regard to promotions and the like. These facts do not stop people from doing interdisciplinary work, though, according to this contributor: Daudet believes that most people do interdisciplinary research; they just aren't accorded official recognition for it.

Some more variations on the theme of the hard facts of life as these manifest themselves with regard to interdisciplinarity were heard from the audience. At times during the discussion much of the responsibility for the difficulties inherent in interdisciplinary work with the humanists when researchers from the humanities are mixed together with researchers from the hard sciences, since the technological and mathematical skills on the part of the humanists are so underdeveloped. That this can, regrettably, be the case was mentioned earlier in this paper. Observations such as these invoke shades of the debate which erupted after C.P. Snow's Rede Lecture at Cambridge from 1959 entitled *The Two Cultures.* Nearly half a century later one senses that there still is some currency in Snow's statement that

> I believe the intellectual life of the whole of western society is increasingly being split into two polar groups. When I say the intellectual life, I mean to include also a large part of our practical life, because I should be the last person to suggest the two at the deepest level can be distinguished. I shall come back to the practical life little later. Two polar groups: at one pole we have the literary intellectuals. . . . at the other, and as the most representative, the physical scientists. Between the two a gulf of mutual incomprehension – sometimes (particularly among the young) hostility and dislike, but most of all lack of understanding. They have a curious distorted image of each other. Their attitudes are so different that, even on the level of emotion, they can't find much common ground [7, pp. 3-4].

Literary intellectuals were, according to Snow, the worst apples in the barrel. Other areas of the humanities were not much better off. As I stated earlier in this piece, however, I believe that particularly with regard to CMMIR there is cause for optimism with respect to a greater dovetailing of traditional humanistic inquiry and inquiry of the more "scientific" sort. Each of the issues raised in 3.2 evinces a hybrid humanities-science nature.[11] The very fact that this article advocating these applications of CMMIR appears in this volume is a sign that there is interest in doing CMMIR research on topics stemming from a very humanistic side of the fence.

Another reason for optimism is the sort of response articulated in an e-mail I received after the panel discussion from Mark Havryliv, a young researcher within CMMIR who – acknowledging that the consequences of a lack of dialog between

[11] Amusingly enough, as I was adding this "Reality Check" section to the manuscript which was used in the "working" proceedings for the conference, I happened upon [1] by Oxford computer scientist Lou Burnard which explicitly deals with similar considerations and employs *The Two Cultures* as a point of departure, but does so almost solely with regard to the use of "humanities computing", as it is called, in order to study literary artifacts.

computer musicians and philosophers/thinkers are deleterious – finds the notion of such dialog to be very important and even odd that it should be a novel idea.

Rodrigo Segnini was another e-mail correspondent who commented on the panel discussion and found interdisciplinarity to be a good thing, but warned against an unintended result of efforts to promote it "where members of one discipline stretch their ideas into other domains, often resulting in work that is useless in either one." His suggestion for future self-regulation is to ask researchers whose work is deemed as being exemplary within interdisciplinary studies to review submissions to journals and conferences.

Lastly, Sølvi Ystad wrote me an e-mail not long after the conclusion of the conference about a CMMIR project for which she received three years of funding. I quote from the attached project summary:

> Towards the meaning of sounds
>
> Why do we easily distinguish a sound produced by a breaking glass from the sound produced by a shock on a metallic structure, although the spectral content of the two sounds is very close? Why do we easily accept the ersatz of a horse's hooves made by a sound effects engineer knocking coco-nuts together? These questions, representing every-day examples both illustrate the complexity and the pragmatism behind the rather unknown concept related to the: *meaning of sounds*. This project aims at establishing relations between the structure of sounds, and their impact on human beings. It is based on an interdisciplinary approach associating acoustics, signal processing, psychoacoustics and cognitive neuroscience. The complementarities of these domains make it possible to address this crucial aspect of sound communication [13].

These issues are central to philosophical discussion of meaning and perception, and point to the ways in which the technologies now at the disposal of CMMIR research permit empirical work on topics which were once only discussable in abstract, speculative terms. A project such as this one is a fine example of interdisciplinary work within CMMIR research which extends into the more traditional areas of humanistic music research.

5 Conclusion

Some pages ago I commented that "It is as if CMMIR as an area of research is unaware of its own potential for reaching out or appealing to areas within the more traditional areas of humanistic music research" on the basis of the way the conference had presented itself on its homepage. By the end of the Pisa conference, things had moved several steps in the right direction. The level-headed and cautionary reservations to which some participants gave voice during and after the panel discussion with regard to the potential pitfalls which should be avoided in interdisciplinary CMMR research are warranted, however. They need to be taken seriously and should contribute to the self-regulation and quality control which must accompany this sort of research.

The humanities face tremendous, but exciting challenges in the new century, as questions as to what it is to be human being become increasingly accessible to empirical study, as is the case with the nature of our perception and memory of music as these may be studied by CMMIR. As a philosopher, I see the opportunity for the direction and development of technology towards the ends of greater understanding of what it is to be human, and – conversely – the embracing of technology on the part of the humanities to be the most exciting future for both fields of endeavor, and CMMIR is one of the paradigm areas where this fruitful exchange is already taking place.

References

1. Burnard, Lou. (1 Dec 2000 (revised Dec 19 2000)) “From Two Cultures to Digital Culture: The Rise of the Digital Demotic.” http://users.ox.ac.uk/~lou/wip/twocults.html
2. Christensen-Dalsgaard, Jakob. (2004) “Music and the Origin of Speeches.” *JMM: The Journal of Music and Meaning* 2, Spring 2004, www.musicandmeaning.net, section 2.*JMM: The Journal of Music and Meaning*, www.musicandmeaning.net
3. Magne ,Cyrille; Mitsuko Aramaki, Corine Astesano, Reyna Leigh Gordon, Sølvi Ystad, Snorre Farner, Richard Kronland-Martinet & Mireille Besson. (Forthcoming) “Comparison of Rhythmic Processing in Language and Music: An Interdisciplinary Approach” *JMM: The Journal of Music and Meaning* 3, Fall/Winter 2004-2005, www.musicandmeaning.net
4. Malloch, Stephen (1999) “Mothers and Infants and Communicative Musicality.” In: *Rhythms, Musical Narrative, and the Origins of Human Communication.* Musicae Scientiae, Special Issue, 1999-2000, pp. 29-57. Liège: European Society for the Cognitive Sciences of Music.
5. Murphy, D. T. H. Andersen, and K. Jensen. (2003) “Conducting audio files via computer vision.” *Proceedings of the Gesture Workshop, Genova.* Springer Verlag, (Available at http://www.diku.dk/~declan/pub/papers.html.)
6. *NTSMB: Netværk for Tværvidenskabelige Studier af Musik og Betydning/Network for Cross-Disciplinary Studies of Music and Meaning,* www.ntmsb,dk, Institute of Philosophy, Education, and the Study of Religions, University of Southern Denmark, Odense Denmark. See specifically programs for *Nature, Culture and Musical Meaning* and *Music, Logic and Technology* both held at SDU in 2002.
7. Snow, C.P (1959/1993) *The Two Cultures.* Cambridge: Cambridge University Press.
8. Tarabella, Leonello (2005) “Handel, a *Free-Hands* Gesture Recognition System.” In: *Computer Music Modeling and Retrieval. Second International Symposium, CMMR 2004. Esbjerg, Denmark, May 2004. Revised Papers,* pp. 139-148. Berlin, Heidelberg: Springer Verlag.
9. Trevarthen, Colwyn. (1994) “Infant Semiosis.” In: Noth, W. (ed.), *Origins of Semiosis.* (pp. 219-252). Berlin: Mouton de Gruyter.
10. Trevarthen, Colwyn. (2001) “Intrinsic Motives for Companionship in Understanding: Their Origin, Development and Significance for Infant Mental Health.” In: *International Journal of Infant Mental Health.* 22 (1-2): 95-131.
11. Trevarthen, Colwyn. (2002) “Origins of Musical Identity: Evidence from Infancy for Musical Social Awareness.” In: MacDonald, R., David J. Hargreaves, D.J. and Dorothy Miell, D. (Eds.) *Musical Identities*, (pp. 21-38). Oxford: Oxford University Press.
12. Whitehouse, Harvey (2000*) Arguments and Icons: Divergent Modes of Religiosity.* Oxford University Press, Oxford
13. Ystad, Sølvi (2005*)* One-page project description entitled “Towards the meaning of sounds.”

Author Index