CONTENT-BASED MELODIC TRANSFORMATIONS OF AUDIO MATERIAL FOR A MUSIC PROCESSING APPLICATION

Emilia Gómez, Gilles Peterschmitt, Xavier Amatriain, Perfecto Herrera

Music Technology Group
Universitat Pompeu Fabra, Barcelona, Spain
{egomez,gpeter,xamat,pherrera}@iua.upf.es

ABSTRACT

This paper presents an application for performing melodic transformations to monophonic audio phrases. The system first extracts a melodic description from the audio. This description is presented to the user and can be stored and loaded in a MPEG-7 based format. A set of high-level transformations can then be applied to the melodic description. These high-level transformations are mapped into a set of low-level signal transformations and then applied to the audio signal. The algorithms for description extraction and audio transformation are also presented.

1. INTRODUCTION

We present here some functionalities of a music content description and processing application that has been developed in the context of the CUIDADO IST project [1]. The prototype, termed *Sound Palette*, is a tool for musicians and audio engineers to automatically generate metadata from the early stages of a production process, and also for using those metadata to control audio transformations. The *Sound Palette* allows the user to operate on audio file much as he/she is used to do on MIDI files, and beyond, as if audio files were "flexible". This idea already appears in the *Melodyne* software [2], which offers the possibility of transforming audio files in a way that was only formerly possible with MIDI. Alongside the usual editing tools (e.g. transformations at note-level) our system incorporates some "macro" functions at different levels (e.g. phrase or motive level), which encapsulate specific musical knowledge; in this way, interesting transformations can be granted to the user using simple controls.

Sound transformations have already been used in a creative environment (see for example the work by the 'Composers' Desktop Project [3], which provides a system to transform sounds for musical purposes). Our goal is to enhance the user/composer creative process by proposing automatically generated musically meaningful material, by means of simple transformation macros. The composer can then select the material of interest for his own composition, which he/she can in turn edit to reach his creative goals (for example by taking advantage of the usual transformation tools provided at note-level). The coupling of a detailed content description with a transformation/re-synthesis algorithm that can be semantically controlled should enable to reach this goal in a way that is intuitive to the user.

2. SYSTEM ARCHITECTURE

The system architecture is presented in Figure 1. First, the audio signal is analyzed, and a set of melodic features is extracted. This description can be stored and loaded from/to a XML document. The melodic transformation chosen by the user is first mapped into a set of modifications of the melodic features (generating a new high-level melodic description). This transformation is then accomplished by a set of low-level audio processing algorithms that are applied to the audio signal in order to generate the desired transformed audio.

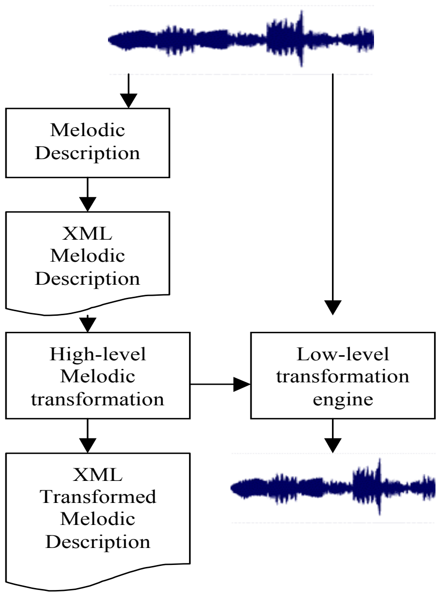
Melodic Description
XML Melodic Description
High-level Melodic transformation
Low-level transformation engine
XML Transformed Melodic Description

Figure 1: System architecture.

3. DESCRIPTION SCHEME

The description scheme that has been used in this work is based on the MPEG-7 standard, incorporating other descriptors needed by the application, as the Note segment temporal location (onset and offset position), note fundamental frequency (not quantized pitch value) and note intensity. The system also stores some global descriptors for the whole phrase. Some of them are changed when performing a high-level transformation (e.g. melodic density transformation for ornamentation & reduction

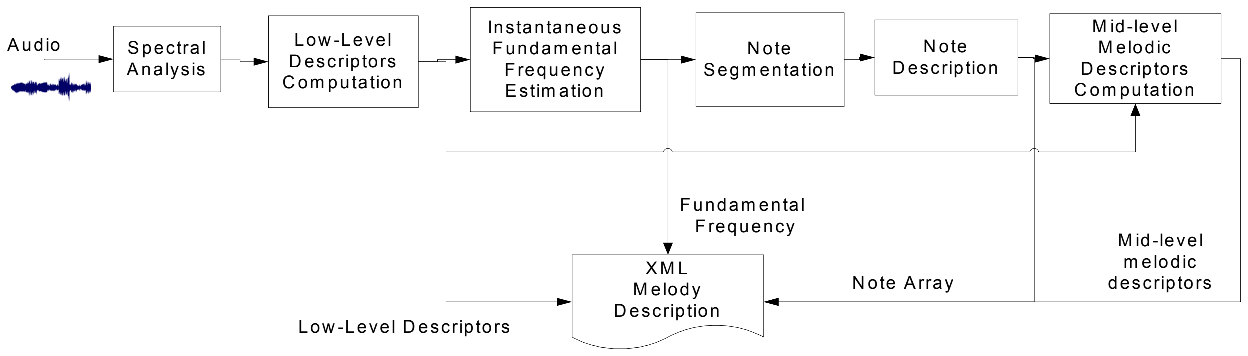
Audio
Spectral Analysis
Low-Level Descriptors Computation
Instantaneous Fundamental Frequency Estimation
Note Segmentation
Note Description
Mid-level Melodic Descriptors Computation
Fundamental Frequency
XML Melody Description
Note Array
Mid-level melodic descriptors
Low-Level Descriptors

Figure 2: Feature Extraction.

transformations) and some of them are used as content information to control and guide transformations [4].

The fact of working with audio material and not with MIDI makes necessary to include in the melodic description scheme some features that are not stored on a MIDI-like melodic representation. Actually, there are descriptors that are lost when converting audio into MIDI that we believe are important when working with audio as, for example, those representing articulation, timbre and expressivity aspects. Although in this first implementation we use mainly note pitch, temporal location and energy information (that is, a MIDI like note representation), the rest of descriptors are intended to give way for implementing more attractive high-level transformations.

4. ALGORITHMS FOR AUDIO PROCESSING

4.1. Algorithms for description extraction

Figure 2 presents a general schema of the melodic description system. Three types of descriptors are extracted; low-level signal descriptors associated to an analysis frame, note descriptors (after note segmentation) associated to a note segment, and global descriptors.

4.1.1. Spectral Analysis

First of all, we perform a spectral analysis of a portion of sound, called analysis frame, whose size can be configured. This spectral analysis consists in multiplying the audio frame with an appropriate analysis window and performing a Discrete Fourier Transform (DFT) to obtain its spectrum. Peak detection is performed on the spectrum and the spectral peaks frequency and amplitude values are determined by parabolic interpolation.

4.1.2. Low-level descriptors

Low-level descriptors are then extracted in frequency domain and attached to each of the analysis frames. Those low-level descriptors are then used by the fundamental frequency and the note estimation algorithms. They are also used to compute the note descriptors.

4.1.3. Fundamental frequency estimation

The fundamental frequency estimation algorithm is based on a harmonic matching model. We implemented a version of the Two-Way mismatch algorithm [5] that computes an estimation of the fundamental frequency from a set of spectral peaks. The spectral peaks are compared to a harmonic series and an error is computed for each of the fundamental frequency candidates. Some extensions and improvements have been implemented in the context of the SMS (Spectral Modeling Synthesis) model [6][7]:

- Peak selection: a peak selection routine has been added in order to eliminate spectral peaks not corresponding to harmonic partials. We use a masking threshold around each of the maximum magnitude peaks. The form of the masking threshold depends on the peak amplitude, and uses three different slopes depending on the frequency distance to the peak frequency.
- Optimization of the fundamental frequency candidates: we only consider candidates some frequencies as the spectral peak ones, the distances between peaks, and those related to them by integer ratios.
- Frequency resolution: the absolute minimum of the error function is found around the final candidate by frequency interpolation, in order to increase the frequency resolution.
- Context awareness: we take into account previous values of the fundamental frequency estimation to perform stability correction and avoid isolated errors. We also have the option to use instrument dependencies to fix the algorithm parameters.
- Noise gate: in the preprocessing step, a noise gate based on some low-level signal descriptors (mainly energy) is applied to detect non-pitched frames.

4.1.4. Note segmentation

For note segmentation, energy onsets are detected following a band-wise algorithm that uses some psychoacoustic knowledge [8]. Pitch changes are detected using fundamental frequency information. Both results are combined to estimate the final note boundaries.

4.1.5. Note descriptors

Once we have computed the note boundaries, note descriptors are computed using this information and low-level frame descriptors. The low-level descriptors associated to a note segment (as e.g. energy, centroid, spectral flatness, etc) are computed by averaging the frame values within this note segment. Fundamental frequency histograms have been used to compute the pitch note and the fundamental frequency that represents each note segment, as found in [9]. This is done to avoid taking into account frames with incorrectly estimations of the fundamental in the computation of its average.

4.1.6. Global descriptors

Global descriptors (that is, attached to the whole phrase) are finally computed using note descriptors and low-level descriptors. These descriptors, labeled "mid-level melodic descriptors" in Figure 2, are numerical descriptors that code the global melodic features of the whole phrase in terms, for instance, of melodic contour, note features distribution (duration, fundamental frequency and interval distribution) or dissonance [7].

4.2. Audio transformation algorithms

4.2.1. Pitch Shifting

The pitch-shifting algorithm is based on a Sinusoidal plus Residual model of the sound and includes a timbre preservation algorithm [10].

A harmonic sinusoidal analysis is performed, which extracts harmonic spectral peaks from the spectrum at multiples of the detected fundamental frequency. The residual spectrum is obtained by subtracting the sine spectrum (synthesised from the windowed-sinusoid model of the spectral peaks) from the original spectrum. The spectral envelope is estimated for each frame from the spectral peaks information. The spectral peaks are simply shifted by the desired pitch-shifting factor, and the original spectral envelope is applied to the new peaks in order to preserve the timbre of the original sound. The residual spectrum is comb-filtered for emphasising harmonic frequencies of the new fundamental frequency. The sinusoidal spectrum is resynthesised from the transformed spectral peaks and added to the transformed residual spectrum. The transformed signal is then reconstructed by IFFT, windowing the resulting signal by a triangular window and finally using the usual overlap-add method [10].

4.2.2. Time Stretch

The algorithm used for time stretch is a frame based frequency domain technique based on the SMS model. The resulting spectral frames are obtained by interpolating both the sinusoidal and the residual component separately on a frame-by-frame basis.

Unlike the phase-vocoder method [10], the synthesis hop size is kept constant, meaning that if the time stretch factor is meant to make the sound slower, new frames will be generated. Thus, the synthesis time will advance at the constant rate specified by the hop size while the pointer to the analysis time will advance according to the time-stretching factor, which can also be time-varying.
These are the basic steps for the algorithm:

- Advance the analysis time pointer according to current time stretching factor.
- Take the nearest analysis frames one to the left and one to the right of current analysis time.
- Interpolate these frames. For computing the interpolation factor use the relation between the current analysis time and each of the frame center time.
- Add the resulting frame as a synthesis frame, using the current synthesis time as its center time.
- Add hop size samples to the synthesis time pointer.

Although the results of the algorithm are not as good as those resulting from the more complex algorithm presented in [11] and [10], its quality can be considered more than acceptable for monophonic phrases.

4.3. High-level description transformation

Formulating transformations at the higher "melodic level" enables to reach melodic transformation at the level of phrases or motives. It is much easier to generate meaningful musical structures at a higher level (phrase, motive level) than it is working at the level of notes (using next-note probability for example) [12]. It is natural in many different musical styles to express thematic development as phrases related to each other through transformations.

The transformations chosen are simple mathematical transformations that are musically meaningful at the higher level and that map naturally to the low-level audio signal transformation. They transform the melodic description according to some parameters. The transformations include:

- Transposition: or global change of pitch.

  $$f_t[n] = \alpha \times f_o[n] \qquad \text{for } n = 1 \ldots N \tag{1}$$

  where $f_o[n]$ is the original pitch value of note n, N the number of notes in the melody and α the pitch transposition factor.
- Horizontal Symmetry: The user can choose a pitch value (arbitrary or some global descriptor related to pitch distribution as minimum, maximum or mean pitch value of the melody) and perform a symmetric transformation of the note pitches with respect to this value on a horizontal axe.

  $$f_t[n] = \exp(2 \cdot \log(f_s) - \log(f_o[n])) \quad \text{for } n = 1 \ldots N \tag{2}$$

  where f_s is the reference pitch value for the symmetry axis.

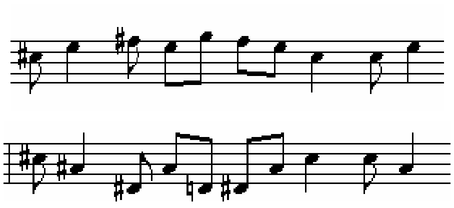
Figure 3: (top) Original melody (bottom) Transformed melody after horizontal symmetry with respect to its minimum pitch value.

- Reverse Pitch (vertical symmetry): it is like traditional reverse, except it only reverses the pitch of the notes [13].

$$f_t[n] = f_o[N-n] \qquad \text{for } n = 0 \ldots N-1 \tag{3}$$

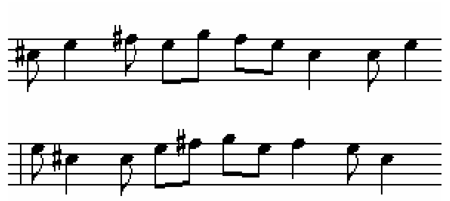
Figure 4: (top) Original melody. (bottom) Reversed-pitch melody.

- Change of interval distribution without changing the global melodic contour. The user is allowed to scale the interval depth by a certain factor (constant or as a function of the note number) without changing the interval direction.

$$I_o[n] = \Delta \times I_i[n] \text{for n} = 1 \ldots N-1 \tag{4}$$

  where $I_o[n]$ is the interval between note *n* and *n+1*, and Δ the interval scaling factor.
- Change of contour direction without changing the interval depths. The user is also allowed to change the interval direction without changing the interval depth (e.g. converting an ascending octave to a descending one).

$$C_o[n] = \beta \times C_i[n] \quad \text{for } n = 1 \ldots N-1 \tag{5}$$

  where $C_o[n]$ is the interval direction between note *n* and note *n-1*, and $\beta \in [-1,1]$.
- Tempo change: the global tempo of the melody can be modified by a user-defined factor.
- Tempo variations: the user can draw a temporal envelope defining the time-scale factor, enabling to apply ritardando and accelerando as desired.
- Reverse time: it is like traditional reverse, except it only reverses the duration of the notes.

$$d_t[n] = d_o[N-n] \quad \text{for } n = 0 \ldots N-1 \tag{6}$$

  where $d_o[n]$ is the original duration of note *n*.

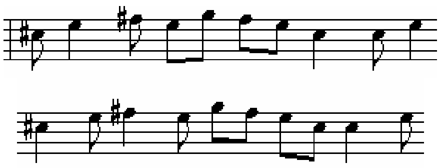
Figure 5: (top) Original melody (bottom) Transformed melody after reversing the note durations.

- Ornamentation and reduction: here, the user is able to modify the "melodic density" descriptor. Ornamentation consists in defining macros for the typical ornamentations such as trill, mordant, cadence, etc, following their definition in music theory textbooks [14]. These macros can be applied in a sensible manner to notes selected by the user.

Each type of high-level transformation affects one descriptor or "attribute" of the melody description (i.e. the intervals, the notes pitch/length...). After each transformation of a given attribute, all the other melodic descriptors are automatically re-computed from the new values of the transformed descriptor. The architecture of the melody description transformation process enables to simply define/program new transformations for one given melodic descriptor without worrying about the subsequent descriptors recalculation.
Although these transformations are conceptually simple, they create dramatic or subtle changes that may enhance the original material according to usual music composition procedures (if used in the right creative context). They have often been used by numerous composers of different musical styles. One key aspect for any high-level melodic transformation macro to be sensible is the context in which it is used. In the *Sound Palette*, the user is able to select the portion of audio (set of notes, phrase...) on which he/she decides to apply the transformation. An interesting functionality of the system is the possibility of applying not only an isolated melodic variation but a set of transformations in chain, multiplying the number of possibilities and resulting in more complex melodic variations.

Finally, these transformations should not be seen as final/definitive variations on a given melody, but as a kind a scratch pad, as a tool to investigate in a fast and intuitive manner the territories lying around a given melody.

4.4. Mapping from high-level transformation to audio processing

As depicted above, the high-level transformations affect the melody *description* only (i.e. the audio is untouched). Once the high-level transformation module has modified the melodic descriptors, the description of the original audio is compared with the transformed description in order to set the parameters of the audio tranformation algorithms. For example, a simple transposition in the melodic description will result in a variation of fundamental frequency descriptor by a certain factor. This factor is extracted from the melodic descriptors comparison and passed to the pitch-shifting algorithm.

As the original frame-by-frame fundamental frequencies values are used as reference for the transformation, the intra-note

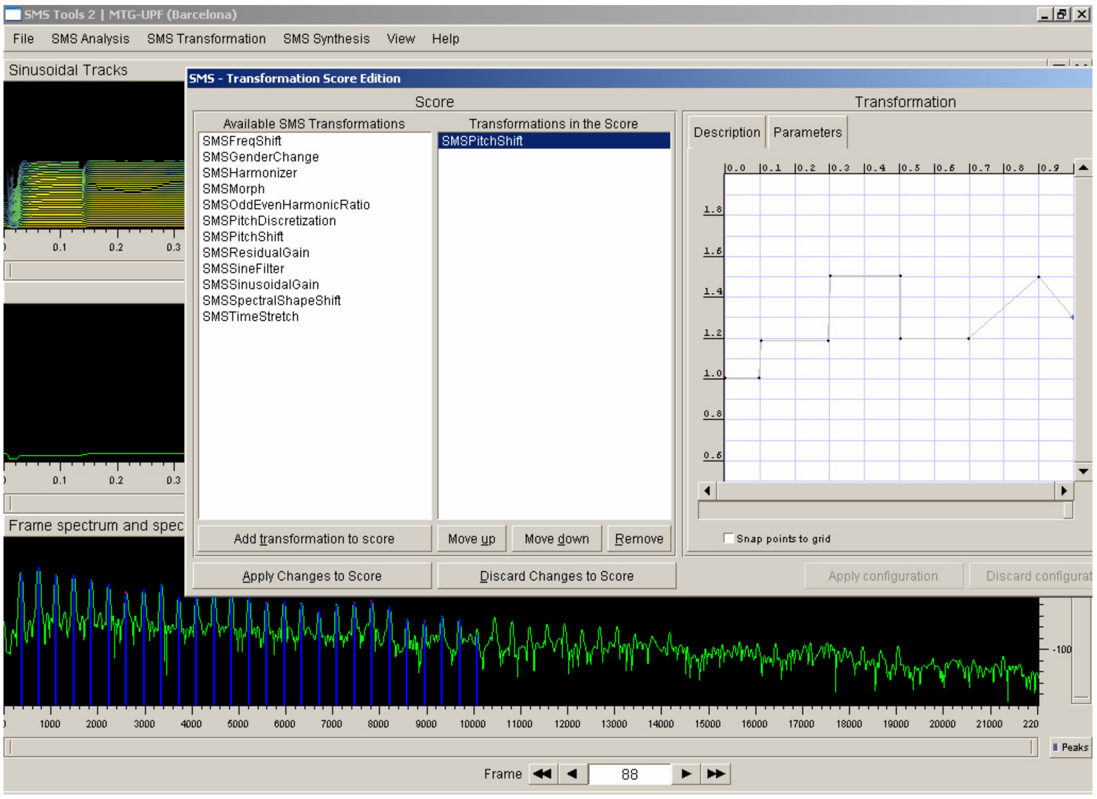

Figure 6: *SMSTools2* application screenshot. Foreground: transformation editor, where different transformations can be added in any desired order, and the transformation parameters can be set with the BPF apparent on the right. Background: sinusoidal tracks, fundamental frequency and spectrum of a given frame with the detected spectral peaks.

expressive variations in fundamental frequency are kept after transformation, conserving some of the expressivity of the original audio. This can be an interesting feature in some cases.

5. IMPLEMENTATION

The system has been implemented in the CLAM (C++ Library for Audio and Music) framework [15][16], which is currently being developed in the MTG and at the time of this writing is on its release 0.5.1.

CLAM is a C++ cross-platform framework for building audio and music applications that is used in most of the MTG's implementations and also has been made public under the GPL license. The framework includes general tools for audio and MIDI input/output, visualization and audio processing in general. But it also implements a general architecture and some ready-to-use C++ classes that are of particular interest to this application. The CLAM framework architecture is mainly built on two general concepts: Processing and Processing Data. Processing classes encapsulate algorithms and offer a convenient and generic way to handle the data and control flow. Processing Data classes offer a tree-like data container structure with automatic support for traversing the structure, memory handling and XML support.

In the CLAM repository a set of Processing classes implement the complete SMS analysis/synthesis scheme as well as the signal level transformations used in the system here presented. It also includes Processing Data for representing audio, spectrum, spectral peaks, frames or segments. Other higher-level Processing Data used in the system, such as the melody representation, have also been implemented using the CLAM architecture, taking advantage, for example, of the automatic XML serialization module.

Finally, the application interface has been implemented using CLAM's Visualization Module and the FLTK graphical toolkit library.

The final application can be run on MS Windows or any GNU/Linux (and will be also available for the Mac OSX platform as soon as the ongoing port of the CLAM framework is finished). Due to the quality of the underlying C++ it can be considered a robust and highly efficient application for melody processing.

6. CONCLUSIONS AND FURTHER DEVELOPMENTS

A system for high-level melodic description and basic transformations has been presented. We proposed a way of helping the composer in his creative process by the use of higher-level description and transformation of melodies.

Improvements are still to be done in the description's robustness and the quality of the final synthesized sound.

For the system to be of real help to the composer, higher flexibility will be added in the transformation macros definition in order to suit his/her particular needs.

A graphical interface for the presentation of the high-level melody description and the editing of the transformations will be implemented and integrated in *SMSTools*. It is crucial for the full-usability of the system that the interface be intuitive and interactive (musician-friendly).

In the future, other transformations that use more complex musical knowledge will be included in the system such as key changes, fitting "equivalent melody" to new chord progression, smoothly transforming one melody to another one. Finally, a very interesting idea would be to add the possibility to transform at the same time the main aspects of sound, that is the melodic, rythmic, timbral and expressive aspects of sound. Transformations can be thought of such as a "Melodic Morph" that would interpolate between different aspects, for example morph both the timbre and melodic contour of the sounds.

7. ACKNOWLEDGEMENTS

The authors would like to thank David Garcia for CLAM support. This research has been carried in the content of the European IST project CUIDADO [1] and the Spanish TIC project TABASCO.

8. REFERENCES

[1] CUIDADO website, http://www.cuidado.mu

[2] Melodyne website, http://www.celemony.com/melodyne

[3] CDP (Composers Desktop Project) website, http://www.bath.ac.uk/~masjpf/CDP/CDP.htm

[4] Gómez, E. Gouyon, F. Herrera, P. Amatriain, X. 2003. *Using and enhancing the current MPEG-7 standard for a music content processing tool*, Proceedings of Audio Engineering Society, 114th Convention. Amsterdam, The Netherlands. http://www.iua.upf.es/mtg.

[5] Maher, R. C and Beauchamp, J. W. 1993 *Fundamental frequency estimation of musical signals using a two-way mismatch procedure*, Journal of the Acoustic Society of America, Vol. 95, page 2254-2263.

[6] Cano, P 1998. Fundamental frequency estimation in the SMS analysis. In *COSTG6 Conference on Digital Audio Effects (DAFX)*, 1998. http://www.iua.upf.es/mtg.

[7] Gómez, E. Klapuri, A. Meudic, B. 2003. *Melody Description and Extraction in the Context of Music Content Processing*, Journal of New Music Research Vol.32 .1.

[8] Klapuri, A. 1999. S*ound Onset Detection by Applying Psychoacoustic Kbowledge,* IEEE International Confernece on Acoustics, Speech and Signal Processing, ICASSP 1999.

[9] McNab, R. J., Smith, L. A. and Witten, I. H. 1996 *Signal processing for melody transcription*. SIG, Working paper, 95(22).

[10] Amatriain, X. Bonada, J. Loscos, A. Serra, X. 2002. '*Spectral Processing'*, Udo Zölzer Ed., DAFX: Digital Audio Effects, p.554, John Wiley & Sons Publishers.

[11] Bonada, J. 2002. *Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio*, Proceedings of International Computer Music Conference, Berlin, Germany. http://www.iua.upf.es/mtg.

[12] Jacob, L. B. 1996. *Algorithmic Composition as a Model of Creativity*, Organised Sound, volume 1, number 3, 1996.

[13] Hinojosa, R. 2003. *Some Projects and Reflections on Algorithmic Music*, Proceedings of Computer Music Modeling and Retrieval, Montpellier, France. http://www.iua.upf.es/mtg.

[14] Blood, B. *Music Theory Online*. http://www.dolmetsch.com/theoryintro.htm

[15] Amatriain, X. de Boer, M. Robledo, E. Garcia, D. 2002. *CLAM: An OO Framework for Developing Audio and Music Applications*, Proceedings of 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Applications. Seattle, WA, USA. http://www.iua.upf.es/mtg.

[16] CLAM website in MTG, http://www.iua.upf.es/mtg/clam