Spectral modeling of musical sounds

Xavier Serra
Audiovisual Institute, Pompeu Fabra University
http://www.iua.upf.es
xserra@iua.upf.es

1. Introduction

Spectral based analysis/synthesis techniques offer powerful tools for the processing of sounds. They include many different approaches, each one with its own advantages and set of compromises that have to be considered before using any of them for a particular application. In this article we present several complementary spectral representations, techniques to obtain them or generate sound from them, and a recently developed graphical interface to better explore their capabilities.

This paper is part of an ongoing research on spectral based techniques for the analysis and synthesis of musical sounds [1][2][3][4]. Throughout all this work the term *Spectral Modeling Synthesis* (SMS) has been used to refer to the software implementations that were first done by Xavier Serra and Julius Smith at Stanford University, and more recently by the Music Technology Group of the Audiovisual Institute of the Pompeu Fabra University in Barcelona. The goal of this work has been to get general and musically meaningful sound representations based on analysis, from which musical parameters can be manipulated while maintaining high quality sound. These techniques can be used for synthesis, processing and coding applications, while some of the intermediate results might also be applied to other music related problems, e.g., sound source separation, musical acoustics, music perception, or performance analysis.

2. Complementary Spectral Models

There is no single spectral representation optimal for all sounds, not even for the different parts of a single complex tone. Thus, we have started by choosing a few basic spectral models that complement each other and can be combined to represent any sound. We are specially interested in techniques suitable for an analysis/synthesis process.

- *Short-Time Fourier Transform (STFT):* This is the most general but least flexible representation and can be used for sounds, or part of sounds, whenever the other models might not give the sound quality desired [6].

- *Sinusoidal:* This is a level of abstraction higher than the *STFT* and it models time-varying spectra as sums of time-varying sinusoids [7]. It is still quite general and there is a gain in flexibility compared with the *STFT*.

- *Sinusoidal plus Residual:* This is a level of abstraction higher than the *Sinusoidal* representation where the sinusoids only model the stable partials of a sound and the residual, or its approximation, models what is left, ideally an stochastic component. It is less general than either the *STFT* or the S*inusoidal* representations but it results in an enormous gain in flexibility [3].

- *High Level Attributes*: From any of the previous representations, specially from the *Sinusoidal plus Residual* model, higher level information such as: pitch, spectral shape, vibrato, or attack characteristics, can be extracted. Depending on the sound more or less information can be obtained. The more we extract the more flexible the resulting representation will be. These attributes will generally accompany one or more of the first three representations [4].

The decision as to what representation to use in a particular situation is not an easy one. Their boundaries are not clear and there are always compromises. The main ones are: (1) sound quality (invertibility), (2) flexibility, (3) generality, (4) memory consumption, and (5) computational requirements. Ideally, in most applications we want to maximize quality, flexibility and generality while minimizing memory consumption and computational requirements. The *STFT* is the best choice for maximum quality, generality and minimum compute time, but the worst one for flexibility and memory consumption. The *Sinusoidal* model is clearly a step towards flexibility by increasing compute time, and the *Sinusoidal plus Residual* model can cover a wide "compromise space" at the expense of a complex and compute intensive analysis process. In fact the *Sinusoidal plus Residual* model is a generalization of both the *STFT* and the *Sinusoidal* models where we can decide what part of the spectral information is modeled as sinusoids and what is left as *STFT*. With a good analysis, the *Sinusoidal plus Residual* representation is very flexible while maintaining a good sound quality, and the memory consumption is quite low. Finally, *High Level Attributes* bring more flexibility to any of the previous models at the expense of increasing the compute time.

For the application of designing a general purpose musical synthesizer [4] our first approach is to try modeling any sound with the *Sinusoidal plus Residual* model and extract as many *High Level Attributes* as possible. When this approach results in an undesirable decrease in sound quality we will use the *Sinusoidal* model, or even the *STFT*. It might be that the attack of a sound is best modeled with the *STFT* while its steady state and release with the *Sinusoidal plus Residual* model.

3. Spectral Analysis

Our particular approach to spectral analysis is based on decomposing a sound into sinusoids plus a spectral residual [3]. This process can be controlled by the user, or done automatically depending on the sound characteristics, and it can produce any of the representations specified above. The analysis procedure detects partials by studying the time-varying spectral characteristics of a sound and represents them with time-varying sinusoids. These partials are then subtracted from the original sound and the remaining residual can be approximated in the frequency domain.

Figure 1 shows a simplified block diagram of the analysis. Considering that we are in the middle of a sound we first prepare the next section to be analyzed by multiplying it with an appropriate analysis window. Its spectrum is obtained by the Fast Fourier Transform (FFT) and the prominent spectral peaks are detected and incorporated into the existing sinusoidal trajectories by means of a peak continuation algorithm. When the sound is pseudo-harmonic, a pitch detection step can improve the analysis by using the pitch information in the peak continuation algorithm and in choosing the size of the analysis window for the next frame. The behaviour of the sinusoids is controlled by the user in such a way that we can either model the whole sound with sinusoids or only the stable partials.

The residual component of the current frame is calculated by first generating the sinusoidal component with additive synthesis, and then subtracting it from the original waveform in the time domain. This is possible because the phases of the original sound are matched and therefore the shape of the time domain waveform preserved. A spectral analysis of this time domain residual is done by first windowing it, window which is independent of the one used to find sinusoids, and thus we are free to choose a different time-frequency compromise. An amplitude correction step can improve the time smearing produced in the sinusoidal subtraction. Then the FFT is computed and the resulting spectrum can be approximated by fitting a curve to the magnitude spectrum. The spectral phases might be discarded when the residual is a stochastic signal.

The result of this part of the analysis is a set of time-varying sines, with or without phase, and a time-varying spectrum, approximated or not. Thus, the output can be: (1) an *STFT*, when no sinusoids have been tracked and the residual spectra are not approximated, (2) a *Sinusoidal* representation, when the tracking of sines has not been too restrictive and no residual obtained, or (3) a *Sinusoidal plus Residual* representation, when the tracking of sines has been restricted to find only stable partials and the spectral residual is calculated, approximated or not.

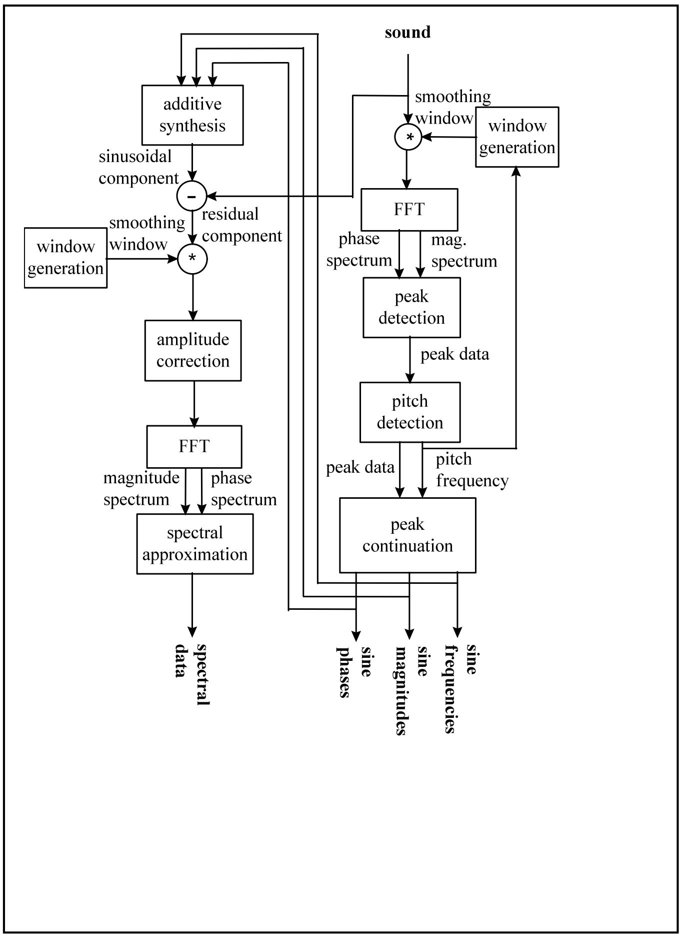

Figure 1: Diagram of the spectral analysis process.

From any of the output spectral information of the analysis we can extract *High Level Attributes* when the sound is a single note or a monophonic phrase of an instrument. Attributes such as attack and release times, formant structure, vibrato, or average pitch and amplitude, can be obtained by the process shown in *Figure 2*. This is quite easy when the spectral representation is of the *Sinusoids plus Residual* type. These attributes can be modified and added back to the spectral representation without any loss of sound quality.

In the case of the spectral representation of a note produced by a musical instrument we first detect the relevant inflexion points for the attack, steady state and release, by studying the amplitude and frequency changes in time. Then the general spectral shape of the steady state of the note is approximated with an envelope and extracted from the whole note. The vibrato of the note is analyzed in the frequency domain, by considering each partial a time function, then computing its time-varying spectra and detecting the spectral peak of the vibrato. The spectral peak is deleted and the inverse-FFT (IFFT) computed, resulting in the initial time-varying partial, now without any vibrato. If this is done both in the frequency and amplitude data we extract both vibrato and tremolo. Then, the overall amplitude and pitch evolution of the note is obtained by averaging all the partials and subtracted from each one. This results in a set of normalized time-varying functions. Finally, we detect the most stable regions of the steady state part of the sound, regions that can be simplified by only storing a few frames of the region and the key frames can also be used as loop points for changing the note duration without affecting the microstructure. The output of this process is the same type of the spectral data that entered it, but now with very little high level information. What is left, if synthesized, does not sound like any instrument, it lacks most of its character, but it has the microstructure that makes the sound “natural”.

The extraction of *High Level Attributes* can be taken a step further by studying and extracting the attributes for an entire instrument, i.e., for all the sounds produced by the instrument. The attributes of each note are compared and combined in order to group the spectral information that is common to the whole instrument, or a part of it, leaving each note only with the differences from the common characteristics of the attributes. This gives musical control at the instrument level without having to access each individual analyzed note.

The approach of comparing and combining *High Level Attributes* is also used to study musical performance characteristics [10], such as the articulation between notes, which are then stored as attributes of the instrument.

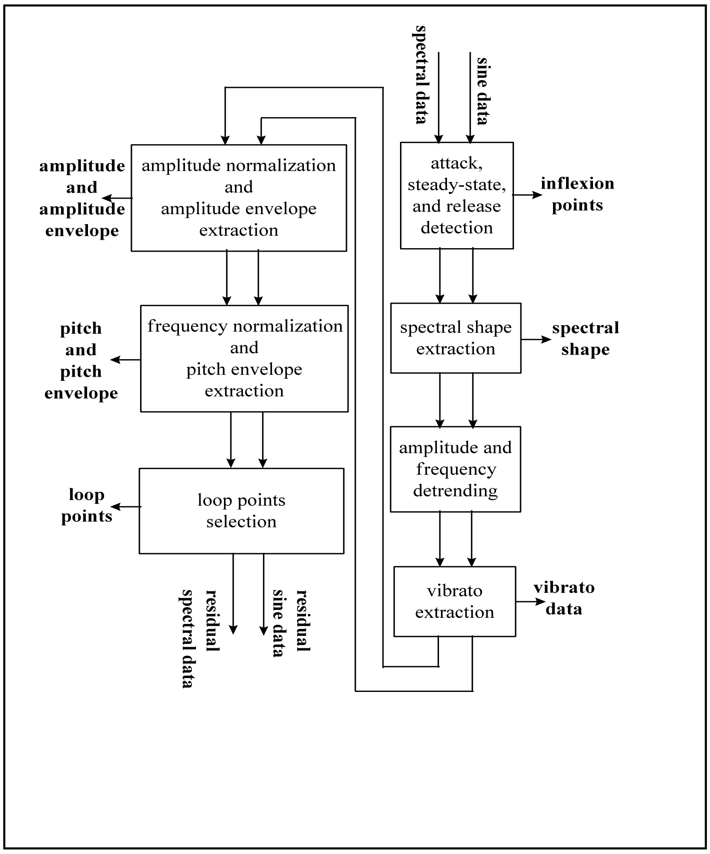

Figure 2: Parameter extraction from spectral data.

4. Spectral Synthesis

The transformation and synthesis of a sound is all done in the frequency domain; generating sinusoids, noise, or arbitrary spectral components, and adding them all, for any number of voices, in a spectral frame. Then, we compute a single IFFT for each frame, which yields very efficient real-time implementations.

Figure 3 shows a block diagram of the final part of the synthesis process. Previous to that we have to transform and add all the *High Level Attributes*, if they have been extracted, and obtain the low level sine and residual data for the frame to be synthesized. Since the stored data might have a different frame rate, or a variable one, we also have to generate the appropriate frame by interpolating the stored ones. The high level control of this process is presented in the next section.

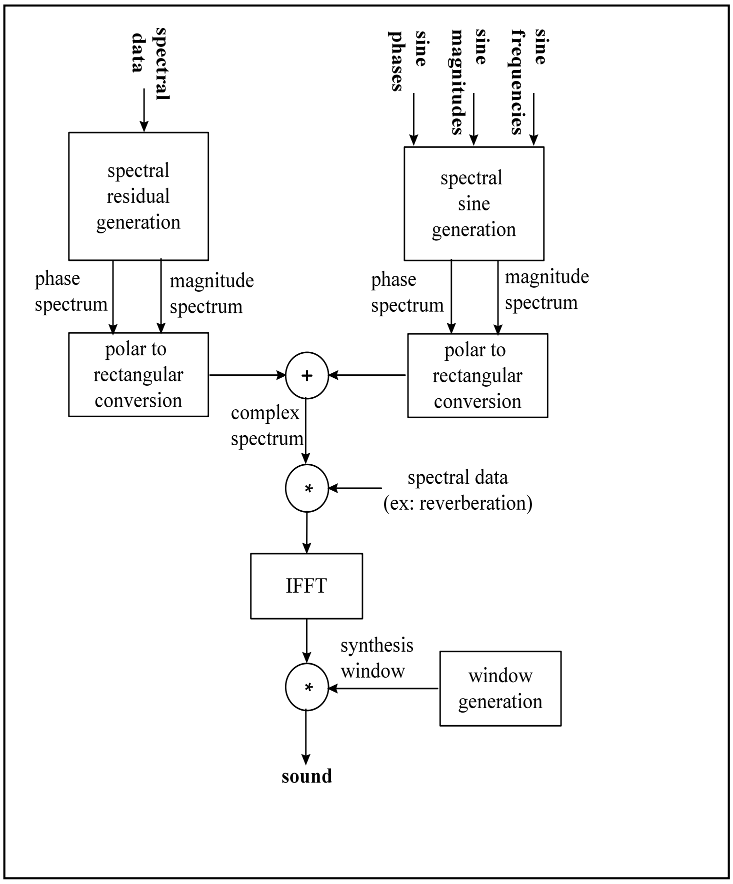

Figure 3: Diagram of the spectral synthesis.

The synthesis of the sinusoids is done in the frequency domain, which is much more efficient than the traditional time domain approach [8]. While it looses some of the flexibility of the oscillator bank implementation, specially the instantaneous control of frequency and magnitude, the gain in speed is significant. This gain is mainly based on the fact that a sinusoid in the frequency domain is a sinc-type function, the transform of the window used, and on these functions not all samples carry the same perceptual weight. To generate a sinusoid in a spectrum, it is sufficient to calculate the samples of the main lobe of the window transform, with the appropriate magnitude, frequency and phase values. We can then synthesize as many sinusoids as we want by adding these main lobes in the spectrum and performing an IFFT to obtain the resulting time-domain signal.

The synthesis frame rate is fixed and completely independent of the analysis one and we would like to have the highest rate possible for maximum control during synthesis. As in all short-time based processes we have to deal with the time-frequency compromise. The window transform, a sine spectrum, should have the fewest possible significant bins since this will be the number of points to generate per sinusoid. A good choice of window is the Blackman-Harris 92dB because its main lobe includes most of the energy. However the problem is that such a window does not overlap perfectly to a constant in the time domain. A solution to this problem is to undo the effect of the window, by dividing by it in the time domain, and applying a triangular window before the overlap-add process [8]. This will give the best time-frequency compromise.

The synthesis of the residual component of the sound is also done in the frequency domain. When the analyzed residual has not been approximated, i.e. it is represented as a magnitude and a phase spectra for each frame, a STFT, each residual spectrum is simply added to the spectrum of the sinusoidal component. But when the residual has been approximated by a magnitude spectral envelope, an appropriate complex spectrum has to be generated.

The synthesis of an stochastic signal from the residual approximation can be understood as the generation of noise that has the frequency and amplitude characteristics described by the spectral magnitude envelopes [3]. The intuitive operation is to filter white noise with these frequency envelopes, that is, performing a time-varying filtering of white noise, which is generally implemented by the time-domain convolution of white noise with the impulse response corresponding to the spectral envelope of a frame. We do it in the frequency domain by creating a magnitude spectrum from the approximated one, or its transformation, and generating a phase spectrum with a random number generator. To avoid periodicity at the frame rate, new random values are generated at each frame.

Once the two spectral components are generated, to add the spectrum of the residual component to one of the sinusoids, we need to worry about windows. In the process of generating the noise spectrum there has not been any smoothing window applied but in the sinusoidal synthesis we have used a Blackman-Harris 92dB, which is undone in the time domain after the IFFT. Therefore we should apply the same window in the noise spectrum before adding it to the sinusoidal spectrum. This is done by convolving the transform of the Blackman-Harris 92dB, only its main lobe since includes most of its energy, by the noise spectrum. This is implemented quite efficiently because it only involves a few bins and the window is symmetric. Then we can use a single IFFT for the combined spectrum. Finally in the time domain we undo the effect of the Blackman-Harris 92dB and impose the triangular window. By an overlap-add process we combine successive frames to get the time-varying characteristics of the sound.

We can take advantage of working in the frequency domain and process the synthesized sound before performing the IFFT. High quality reverberation is very efficiently incorporated in the frequency domain by multiplying stored spectra of impulse responses of reverberations for each synthesized spectral frame. This limits the length of reverberations to the frame size. Longer reverberations can also be applied by splitting their impulse responses into several spectra and doing an appropriate overlap-add over the result.

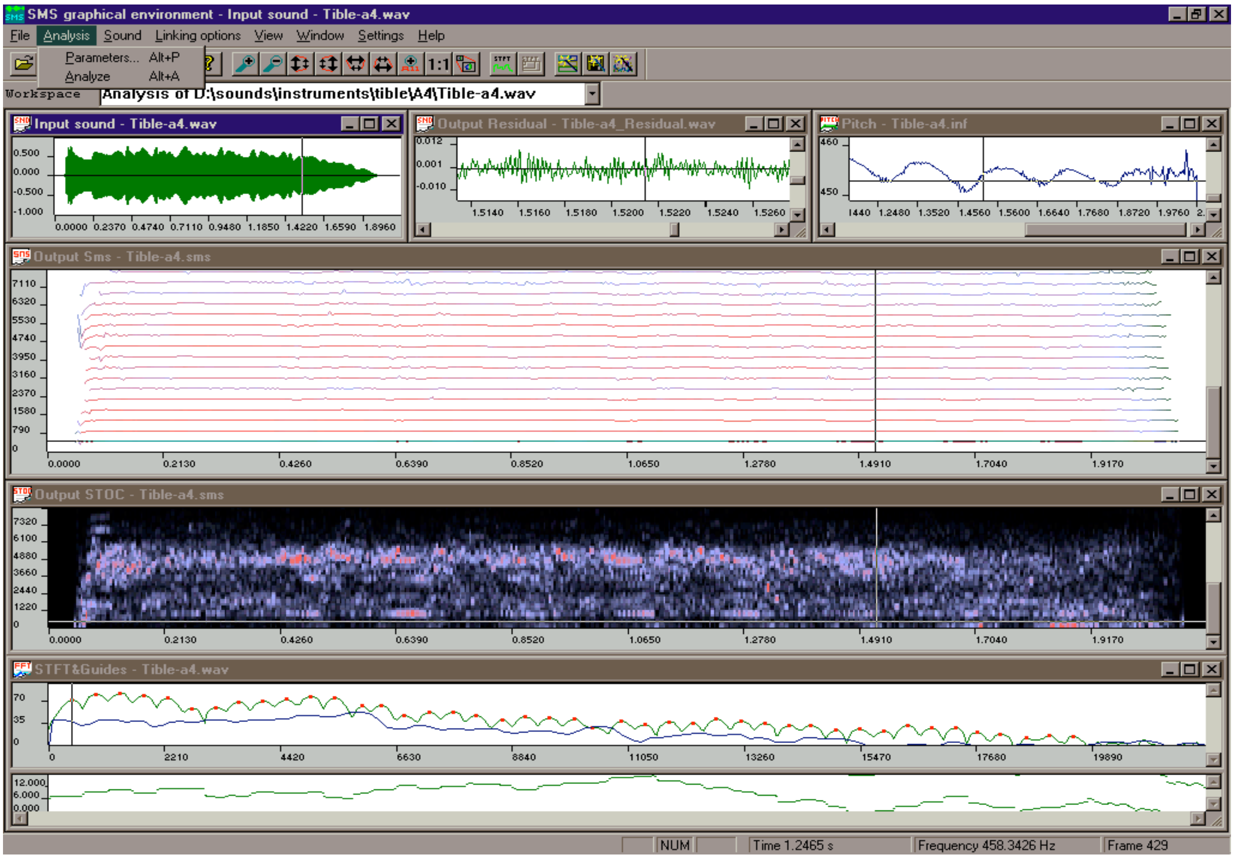

Figure 4: Analysis workspace of the SMS graphical environment.

5. A Graphical Environment

The current implementation of our SMS system has been developed in Visual C++ under Windows 95. It has a graphical front-end that is useful for the exploration of most of its capabilities.

The graphical interface includes three types of workspaces: (1) *analysis*, (2) *transformation-synthesis*, and (3) *event-list control*. With the *analysis* workspace (Figure 4) the user views an original sound, with its time-varying spectrum, and sets the different analysis parameters with menus and graphical tools. Once analyzed, it displays the resulting representation of the sinusoidal and residual components of the sound and the intermediate steps that are useful for understanding the analysis process and finding the optimal analysis parameters. In particular the user can study the pitch detection, partial tracking, and residual approximation processes, steps that are critical for a good resynthesis. All the representations are synchronized and are easily scaleable.

The *transformation-synthesis* workspace is used for the manipulation of the analysis data and the generation of a synthesized sound from it. The user can set all the transformation parameters with different types of menus and graphical drawing tools. These transformations go from manipulations of the different components of the sound (partials and residual) to the hybridization of two analyzed sounds.

The *event-list control* workspace permits to control the transformation and synthesis of several sounds by using a specifically developed *Score File*. The parameters for the different events can either be specified with a text editor or graphical tools, and we can go from one representation to the other and the program translates the data accordingly. With this type of control we can go beyond the transformation and synthesis of single sounds and use most of the potential of the musical synthesizer software [4]. We can define *Collections* of spectral data, (multidimensional spaces of sounds), or use existing ones, and synthesize from them by giving the coordinates of the synthesized sound to be produced and the way to interpolate from the existing analyzed sounds.

We are currently working on workspaces for controlling the extraction of *High Level Attributes* and organizing the analysis of the different sounds produced by an instrument into a *Collection* which is then stored into a single Spectral Description file.

6. Conclusion

In this article we have discussed the use of complementary spectral representations in the analysis and synthesis of musical sounds. Such an approach takes advantage of the qualities of each model and offers the possibility of representing sounds at different levels of abstraction. An implementation of these techniques and the graphical interface described are publicly available on our web site [5].

A part from the applications of sound synthesis and processing we have started using these techniques in other music applications [9][10].

References

[1] Serra, X. 1989. *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition*. Ph.D. Dissertation, Stanford University.

[2] Serra, X. and J. Smith. 1990. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition'', *Computer Music Journal* 14(4):12-24.

[3] Serra, X. 1996. "Musical Sound Modeling with Sinusoids plus Noise", in G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors, *Musical Signal Processing*. Swets & Zeitlinger Publishers.

[4] Serra, X., J. Bonada, P. Herrera and R. Loureiro. 1997. "Integrating Complementary Spectral Models in the Design of a Musical Synthesizer", *Proceedings of the ICMC 97*.

[5] SMS Web site. URL:http://www.iua.upf.es/~sms.

[6] Allen, J.B. 1977. "Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform", *IEEE Transactions on Acoustics, Speech and Signal Processing*, 25(3):235-238.

[7] McAulay, R.J. and T.F. Quatieri. 1986. "Speech Analysis/Synthesis based on a Sinusoidal Representation", *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):744-754.

[8] Rodet, X. and P. Depalle. 1992. "Spectral Envelopes and Inverse FFT Synthesis", *93rd Convention of the Audio Engineering Society*. San Francisco, October 1992.

[9] Loureiro, R. and X. Serra. 1997. "A Web Interface for a Sound Database and Processing System", *Proceedings of the ICMC 97*.

[10] Àrcos, J.L; R. López de Mántaras and X. Serra. 1997. "Saxex: a Case-Based Reasoning System for Generating Expressive Musical Performances", *Proceedings of the ICMC 97*.