ReSpect: A Free Software Library for Spectral Sound Synthesis

Sylvain Marchand

▶ **To cite this version:**

Sylvain Marchand. ReSpect: A Free Software Library for Spectral Sound Synthesis. Proceedings of the Journées d'Informatique Musicale (JIM07), Apr 2007, France. pp.33–43. hal-00308037

HAL Id: hal-00308037
https://hal.science/hal-00308037

Submitted on 4 Nov 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RESPECT: A FREE SOFTWARE LIBRARY FOR SPECTRAL SOUND SYNTHESIS

Sylvain Marchand
SCRIME / LaBRI – CNRS, Université Bordeaux 1
351 cours de la Libération, 33405 Talence cedex, France

Abstract

ReSpect is a free software library performing the real-time synthesis of spectral sounds. The origin of this library is an additive synthesis module, implemented using software oscillators and presented in Paris at the JIM'99 conference. On the top of this efficient synthesis module were added high-level functionalities such as a psychoacoustic model and a spatialization system, all working in a spectral representation. ReSpect accepts several spectral sound models as input. Thus, this library can be used for research in the spectral modeling field, as well as for teaching purposes, since the sources are available. Moreover, its modular approach allows to enable or disable a specific module, which is extremely useful for pedagogy. Its final goal is to serve as a software tool to be included in bigger projects. The ReSpect library is used for example in the Dolabip project, a multi-field project for developing early-learning games for electro-acoustic music at school. It is also used in other computer music software tools of the SCRIME, more oriented towards the composition of electro-acoustic music.

1. INTRODUCTION

The original ReSpect ("Re(spect) Spect(rum)") project started in year 1997. It was implemented in 1998–1999 as a GNU/Linux kernel module [1], and was performing the additive synthesis of spectral sounds in a very efficient way, using very fast software oscillators presented at the JIM'99 conference [2].

At this time, it was only an efficient synthesis module, but without any high-level functionality. This original module can now be regarded as the synthesis (SYN) module of the actual library (Figure 1).

Then the SAS (Structured Additive Synthesis) [3, 4] library was designed in year 2000 on the top of ReSpect in order to play SAS sound sources. This library is used for example in the Dolabip project [5], a multi-field project for developing early-learning games for electro-acoustic music at school.

A psychoacoustic model (PSY) was added to the library in 2001 [6]. A resampling module (RS) was added in 2002. The spatialization (SPA) module was initiated the same year. From 2003–2004, the library was not seriously maintained, although the spatialization module got enhanced by student projects.

The development has started again in 2006, and the new version should be released in 2007. But for now, only the last version of the SAS library is available for download [1], with limited implementations of the spatialization (SPA) and synthesis (SYN) modules, and no event manager (EM) module, see below.

Now, the ReSpect project is back to its original name, and is now a free software library – distributed under the terms of the GNU General Public License – that accepts both SAS (structured additive synthesis) or spectral (classic additive synthesis) frames as input, and generates temporal frames as output to the sound card. More precisely:

1. Partials are extracted from either SAS or spectral (additive synthesis) frames;
2. Once the partials are known, their parameters are resampled by the resampling (RS) module, see Section 3;
3. Each sound source can be played at any position, using the spatialization (SPA) module, see Section 6;
4. The parameters of the oscillators are updated using an event manager (EM) module, see Section 3;
5. A psychoacoustic (PSY) module then decides which partial is audible and must be synthesized, see Section 5;
6. In the end, the additive synthesis (SYN) module computes the sound wave associated to the partials, in a very efficient way, see Section 4.

An important feature of the ReSpect library is that any of these modules can be disabled, except SAS and SYN. This is extremely useful for teaching purposes (*e.g.* for showing the psychoacoustic model), and also to save computation time (*e.g.* for applications where spatialization is not necessary).

The end-user functionalities are very simple, and can be summarized by: playing spectral sounds at any position, in a very efficient way. Thus ReSpect can be useful for any application where real-time additive synthesis is needed. The library API is also

[1] `http://www.scrime.u-bordeaux.fr/logiciels/libsas`

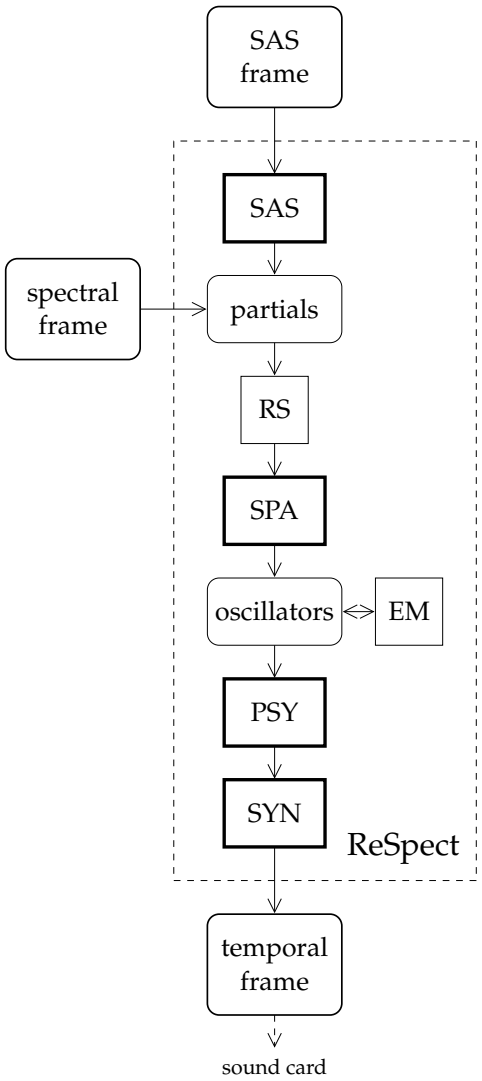

Figure 1. Diagram of the ReSpect library.

very simple, but not detailed here since this API will be part of the user manual of the new version of ReSpect, to be released soon.

After a brief introduction to the spectral and SAS sound models in Section 2, we give some technical insights into the high-level functionalities of the library. With these insights and the source code, computer scientists should be able to fully understand the software library. First of all, the way ReSpect handles the sound parameters is rather original, and is thus detailed in Section 3. Then, we describe the three main modules dealing with fast additive synthesis (SYN), psychoacoustics (PSY), and spatialization (SPA) in Sections 4, 5, and 6, respectively.

2. SOUND MODELS

Spectral sound models provide general representations for sound well-suited for intuitive and expressive musical transformations (see [7, 8, 4]).

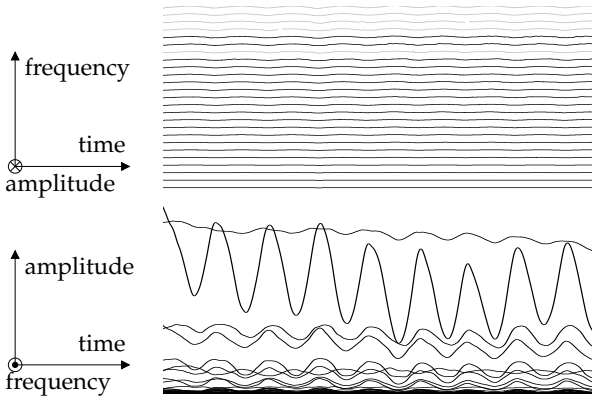

Figure 2. The frequencies (top) and amplitudes (bottom), as functions of time, of the partials of the sound of an alto saxophone, during 1 second.

2.1. Additive Synthesis

Additive synthesis (see [9]) is the original spectrum modeling technique. It is rooted in Fourier's theorem, which states that any periodic function can be modeled as a sum of sinusoids at various amplitudes and harmonic frequencies. For stationary pseudo-periodic sounds, these amplitudes and frequencies continuously evolve slowly with time, controlling a set of pseudo-sinusoidal oscillators commonly called *partials*. This is the well-known McAulay-Quatieri representation [10] for speech signals, also used by Serra and Smith [7, 11] in the context of musical signals. The audio signal s can be calculated from the additive parameters using Equations 1 and 2:

$$s(t) = \sum_{p=1}^{P} a_p(t)\ \sin(\phi_p(t)) \quad (1)$$

$$\phi_p(t) = \phi_p(0) + 2\pi \int_0^t f_p(u)\ du \quad (2)$$

where P is the number of partials and the functions f_p, a_p, and ϕ_p are the instantaneous frequency, amplitude, and phase of the p-th partial, respectively. The P pairs (f_p, a_p) are the parameters of the additive model, and represent points in the frequency-amplitude plane (see Figure 3). This representation is used in many analysis / synthesis programs such as AudioSculpt [12], SMS [11], or InSpect [1].

2.2. Structured Additive Synthesis

However, the models based on additive synthesis are extremely difficult to use directly for creating and editing realistic sounds. The reason for this difficulty is the huge number of model parameters – controlling many sinusoidal oscillators – which are physically valid but only remotely related to musical parameters as perceived by a listener.

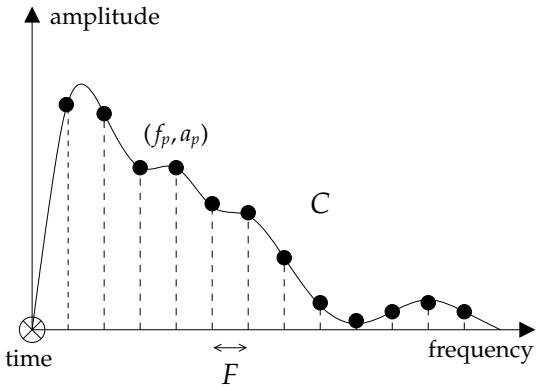

Figure 3. Spectrum of an harmonic sound at a given time t. The sound consists of 12 partials (dots, with dashed lines). The frequency F and color C (solid line) parameters of the SAS model are also indicated.

Structured Additive Synthesis (SAS) [3, 4] is a spectral sound model that keeps most of the flexibility of additive synthesis while addressing these problems. It imposes constraints on the additive parameters, giving birth to structured parameters as close to perception and musical terminology as possible, thus reintroducing a perceptive and musical consistency back into the model.

SAS consists of a complete abstraction of sounds according to only four physical parameters, functions closely related to perception. We note (A, F, C, W) a sound in the SAS model. At a given time t, these parameters constitute an SAS frame (see Figure 1). The first two parameters – amplitude A and frequency F – are one-dimensional, functions of time only, while the two others – color C and warping W – are two-dimensional, functions of both frequency and time.

In the ReSpect library, the A and F parameters are scalars, whereas C and W are envelopes. The SAS module of the library (see Figure 1) generates the partials from the SAS parameters. The number of partials to generate in the spectrum range is:

$$P(t) = \frac{F_s}{2F(t)} \tag{3}$$

where F_s is the sampling frequency.

The main difficulty for the SAS module is then to kill old partials (if P has decreased) or to give birth to new partials (if P has increased), while respecting the protocol for the spectral frames (see Section 3). Indeed, computing the additive parameters for the P partials from the SAS parameters is as easy as:

$$f_p(t) = W(pF(t), t) \tag{4}$$

$$a_p(t) = A(t)\, C(f_p(t), t). \tag{5}$$

3. CONTROLLING THE SYNTHESIS

The synthesis parameters are slow-varying functions of time. More precisely, these functions can be regarded as (control) signals which are band-limited in frequency with a maximal frequency under the lowest audible frequency, that is ≈ 20 Hz (see [8]).

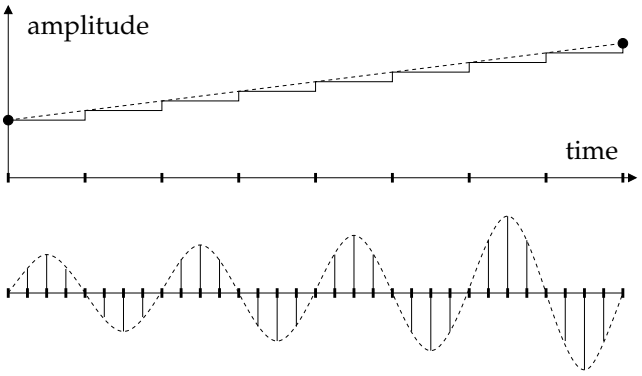

Figure 4. Variation of the amplitude of an oscillator and the resulting audio signal. Between two parameter changes, some interpolated values are computed (8 on the top of this figure). And between two interpolated values, many samples are computed (4 on the bottom of this figure, but 64 in ReSpect). . .

We do not assume a linear variation of the parameters. We intend to do even better to obtain a high quality with a low rate for the parameter flow. Considering the parameter values as the samples of a band-limited signal allows us to reduce the frequency of the updates of the parameter flow.

Since the maximal frequency is under 20 Hz, in theory sampling the parameters 40 times per second or so is sufficient (Nyquist's condition). In practice we use a value of about 86 times per second, in order to be able to encode fast variations for partials with high frequencies. Psychoacoustic experiments indeed show that the maximal frequency of the control signal is not a constant, but a function proportional to the frequency of the controlled signal. More precisely, to avoid modulation phenomena, the maximal frequency of the control signal is about 0.35% of the frequency of the controlled partial.

The default sampling rate of ReSpect is F_s = 44100 Hz (CD quality). ReSpect then asks for sound parameters each 512 samples (thus at ≈ 86 times per second). This is sufficiently fast for the correct handling the control signals, and sufficiently slow not too overload the system by a high parameter flow. However, the period between two parameter changes (512 samples) is too large for the parameters to be considered as constant within this time interval. ReSpect then computes intermediate parameter values, every 64 samples. The consequence is an upsampling of the control signals by a factor 8 (see [13] and Figure 4). Then, during the 64 samples, the sound is considered as stationary (constant parameters) and the fastest synthesis algorithms can be applied, without considering any change of parameters.

The current version of ReSpect is more flexible and has an event manager (EM) module, which uses an

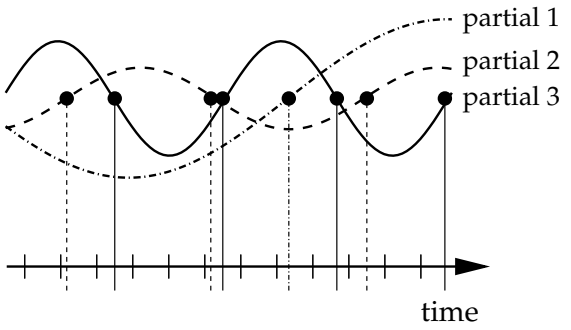

Figure 5. Update events for 3 partials (dash-dotted, dashed, and solid lines). For each partial, they occur here at each $p = 1/2$ period of the sinusoid of the partial, possibly between two output samples.

optimized priority queue (see [14]) and allows the oscillators to take specific actions at scheduled times. Between two events, no parameter change can occur, and the synthesis algorithms can generate the output samples in a very efficient way. Events are attached to oscillators, and there are events of different kinds:

- change of synthesis parameter (see Figure 4), possibly at optimal times (see Figures 6 and 7);
- update of the polynomial generator (every 1/2 period, see Figure 5 and Section 4);
- reset of the digital resonator or incremental generator (depending on the synthesis method, see Section 4), to avoid numerical imprecision and drifts.

This section then details three important points about the update of the synthesis parameters. The first point is the way spectral frames (see Figure 1) must be input. The second point is the way the synthesis parameters can be computed at any time, and not necessarily at times where they were input by the user. The third point shows that is then possible to change the parameters of the oscillators at strategic times, to avoid discontinuities (clicks).

3.1. Spectral Frame Protocol

For each spectral source (additive synthesis, see Section 2), ReSpect maintains an ordered list of the active partials currently synthesized. Each partial p is represented by a pair of floating point numbers in double precision, (f_p, a_p). Recall that f_p and a_p are respectively the frequency and amplitude of the p-th partial. The number of synthesized partials can vary with time. When a partial dies, the special $(0, 0)$ pair is placed at its position in the list of partials. After that, the partial does not exist anymore. When a new partial appears, its is simply appended to the list of partials. When the pairs are send to ReSpect as input, the following protocol must be respected in sequence:

1. For each partial p, write its (f_p, a_p) pair. If a partial dies, the $(0, 0)$ pair is written;
2. For each new partial, write its pair;
3. Write the $(-1, -1)$ end-marker pair.

This steps should be repeated for each frame.

This protocol applies to spectral frames only. For SAS frames, the user inputs the parameters in a more natural way, and then the SAS module has to deal with this protocol when generating the partials.

3.2. Resampling the Parameters

The control parameters are sent only every 512 samples of the output signal. However, synthesis parameter change events can occur at any time. Thus, we must be able to reconstruct the parameter value at any time, and we do this in the RS module (see Figure 1) by reconstruction of the control signal.

As mentioned in [15], this reconstruction should be done in theory using a convolution with a reconstruction filter whose impulse response would be a (windowed) sinus cardinal – sinc – function. Using a convolution would severely degrade the performance of our synthesis algorithm. We propose instead to use an approximation of the ideal reconstruction, and more precisely to use cardinal splines.

The principle of the splines is to use polynomials to approximate or interpolate functions. We are interested in interpolation splines for uniform reconstruction. The cardinal splines are interpolating uniform splines widely used in computer graphics [16]. The s signal can be locally reconstructed from its samples using equation:

$$s(kT_s + t) = \sum_{i=0}^{3} c_i(t)\, s[k - 1 + i] \quad (0 \le t < 1) \qquad (6)$$

where T_s is the sampling period of the (control) signal – indicating times where the signal values are know – and the c_i functions are, for cubic splines, the following Hermite blending functions:

$$c_0(t) = \frac{1}{2}(-t + 2t^2 - t^3) \qquad (7)$$

$$c_1(t) = \frac{1}{2}(2 - 5t^2 + 3t^3) \qquad (8)$$

$$c_2(t) = \frac{1}{2}(t + 4t^2 - 3t^3) \qquad (9)$$

$$c_3(t) = \frac{1}{2}(-t^2 + t^3) \qquad (10)$$

Cubic cardinal splines in conjunction to the well-known Horner method for the evaluation of the polynomials result in a very efficient resampling technique with a sufficient precision. This reconstruction turns out to be of very good quality. The reason for this quality is that these Hermite functions constitute indeed a piecewise-polynomial approximation of the impulse response of a reconstruction filter (see [8]).

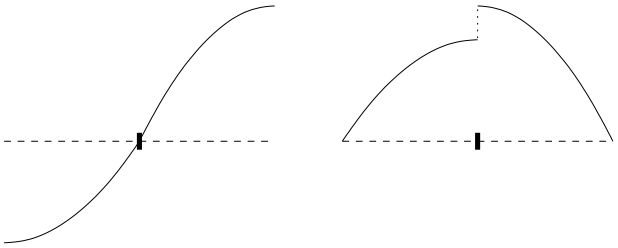

Figure 6. Changing the amplitude either when the signal is minimal (left) or maximal (right). It appears that the left case is much better, since it avoids amplitude discontinuities (clicks).

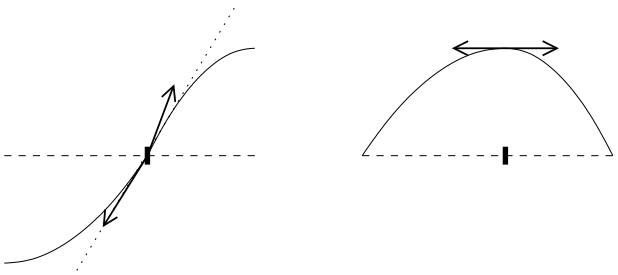

Figure 7. Changing the frequency either when the signal is minimal (left) or maximal (right). It appears that the right case is better, since it avoids derivative discontinuities (clicks).

3.3. Avoiding Discontinuities

In [2] we indicate the best times in a period to change the parameters of a partial, as illustrated by Figures 6 and 7. The best moment to change the amplitude is when the signal is minimal, to preserve the continuity of the signal. And the best moment to change the frequency is when the signal is maximal, to preserve the continuity of the signal derivative. The parameters are thus updated at the right moment to avoid clicks in the sound. This moment is different for each partial, depending on its frequency.

4. FAST ADDITIVE SYNTHESIS

In order to efficiently synthesize many sinusoids simultaneously, Freed, Rodet, and Depalle propose in [17, 18] to use the inverse Fourier transform, provided that the control parameters vary extremely slowly. The gain in complexity is when the number of oscillators is large in comparison to the number of samples to compute at each frame. However, the control of the synthesis parameters lacks suppleness. On the contrary, the ReSpect library is oscillator-oriented to allow a fine control of the synthesis parameters over time (see Section 3).

The most straightforward way to calculate a partial contribution is then to apply Equation 1, using the sine function. But it consumes a lot of computation time. Other techniques are possible.

4.1. Digital Resonators

The digital resonator method [19, 20] computes the samples of each separate partial with an optimal number of operations. In this method, the sinusoidal function is calculated with an incremental algorithm that avoids computing the sine function for every sample. We proposed the use for fast additive synthesis of the digital resonator with floating point arithmetic in [1, 2]. For each partial the resonator is initialized as Equation 12 shows, with F_s the sampling rate of the synthesis, a, f, and ϕ respectively the amplitude, frequency, and initial phase of the partial, and Δ_ϕ the phase increment. The incremental computation of each oscillator sample requires only 1 multiplication and 1 addition:

$$s[n+1] = C \cdot s[n] - s[n-1] \tag{11}$$

$$\text{with} \begin{cases} \Delta_\phi &= 2\pi f / F_s \\ C &= 2\cos(\Delta_\phi) \\ s[0] &= a\sin(\phi_0) \\ s[1] &= a\sin(\phi_0 + \Delta_\phi) \end{cases} \tag{12}$$

This algorithm is optimal in a sense that 1 multiplication with no addition will lead to a geometric progression, whereas no multiplication with 1 addition will lead to an arithmetic one; none of these progressions being a sine function.

Although the complexity of the digital resonator method is proportional to $P \times F_s$, in practice it can compete with the FFT^{-1} [21] or even be more efficient [8], depending on the implementation details and optimizations, on the computer used, as well as the number of partials P.

Numerical stability can be a problem for low-frequency and low-amplitude partials. In practice, the low-amplitude partials are not problematic since they are inaudible and thus removed by the psychoacoustic model (see Section 5).

For low-frequency partials, we propose to use another method, more stable and even more efficient.

4.2. Polynomial Generator

We propose in [14] a new synthesis method whose computation time depends mainly on the mean of the frequencies of the partials. Perhaps surprisingly, we have shown that the complexity of this method does not really depend either on the number of the partials or on the sampling frequency, but rather on the mean of the frequencies of the partials. As a consequence, this method is particularly efficient for low-frequency partials.

The idea is to replace all the P sine functions of Equation 1 by a single polynomial generator of degree d. Our method consists in first calculating a set of polynomial coefficients for each partial. The values from polynomials computed with these coefficients approximate the signal of the partial on a part p of its period. The classic approach would evaluate the polynomial associated to each oscillator, and

then sum up the results, which would be quite inefficient (far less efficient than the digital resonators, see above). The idea is yet to sum the coefficients in a polynomial generator, then to evaluate the resulting polynomial only once. Indeed, summing polynomials leads to another polynomial of the same degree. The sound samples can be computed from this single resulting polynomial, with a fairly low degree – independent of the number of partials to synthesize.

4.2.1. Partial Approximation

The time-domain signal generated by each partial is defined by a sine function. We propose to approximate this function by a polynomial. To get the polynomial coefficients that can approximate any partial of a sound, we decide to first approximate a unit (sinusoidal) signal u with amplitude $a = 1$, frequency $f = 1$, and phase $\phi = 0$.

We have to choose a part of the period where we will do the approximation. We call this part the validity period p of the polynomial coefficients. Thus, if we approximate a half period of u, then $p = 1/2$.

For a given polynomial degree d, we propose to find the polynomial coefficients that maximize the Signal-to-Noise Ratio (SNR) between the target unit signal u and its polynomial approximation U, the noise being the approximation error. These polynomial coefficients also have to respect other constraints to maintain a piecewise continuity. For example, with a 2-degree polynomial U and $p = 1/2$, as well as a C^1 continuity, we show that we can use alternately U_1 for a first half period and U_2 for the second:

$$\begin{cases} U_1(t) = a_1 t + a_2 t^2 \\ U_2(t) = -a_1 t - a_2 t^2 \end{cases} \tag{13}$$

$$\text{with} \begin{cases} a_1 = 240/\pi^3 \\ a_2 = -480/\pi^3 \end{cases} \tag{14}$$

Also note that, in this case, switching from U_1 to U_2 is very efficient.

The coefficients we compute define a unit polynomial U by validity period. When the unit polynomial is found, every partial can be approximated from it. In the general case of a partial p with amplitude a_p, frequency f_p, and initial phase ϕ_p, the approximating polynomial P_p is then given by:

$$P_p(t) = a_p U\left(f_p t + \frac{\phi_p}{2\pi}\right) \tag{15}$$

4.2.2. Incremental Evaluation

To avoid the problem of computing a polynomial with large time values, leading to numerical imprecision, we propose to use the Taylor's theorem to compute it. The polynomial can be evaluated at every instant $t_0 + \Delta_t$ by using its value and the values of its derivatives at a preceding instant t_0:

$$P^k(t_0 + \Delta_t) = P^k(t_0) + \sum_{i=k+1}^{d} \frac{\Delta_t^{i-k}}{(i-k)!} P^i(t_0) \tag{16}$$

where P^k is the k-th derivative of the polynomial function (P^0 being the polynomial itself). The number of necessary values depends on the degree of the polynomial (*e.g.* three values with a 2-degree polynomial).

We have to directly compute the first value of the polynomial and of its derivatives. Then, to compute incrementally each of the following values, we use Equation 16 with a step Δ_t corresponding to the time between two time events. A time event is either the time of a sound sample or of a scheduled update of the coefficients. When time reaches or exceeds the validity period we have chosen, the coefficients are updated, and the incremental algorithm goes on with the new coefficients.

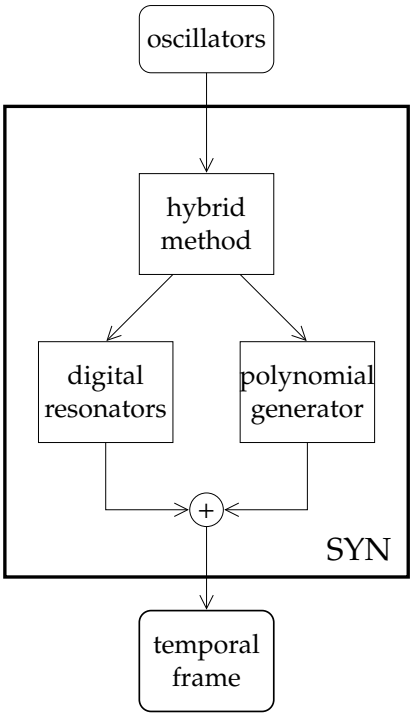

Figure 8. Diagram of the synthesis (SYN) module.

4.3. Hybrid Method

In the ReSpect library, we tune the trade-off between complexity and stability for our synthesis method on the fly, by combining the advantages of the digital resonator and polynomial generator methods, since these methods both manipulate oscillators. For this hybrid method, the idea is, for a given partial, to use either the digital resonator or the polynomial approximation depending on the frequency of the partial. For low frequencies, the polynomial generator will

be preferred. On the contrary, for high frequencies, the digital resonator will be used.

The hybrid method schedules update times as events in the event manager (EM) module (see Figure 1 and Section 3). Whereas the digital resonators only need events for amplitude and frequency changes (that can be scheduled at optimal times, see Section 3), the polynomial generator requires the update times of the polynomial approximations of the partials, scheduled when the validity period p of an oscillator is ending.

Changing parameters of partials with the polynomial method consists in changing the polynomial coefficients of an oscillator when this oscillator must be normally updated, because of the end of its validity period. Thus, this change does not need more computation time than without changing the parameters. The best case is with the validity period $p = 1/4$. In this case we can update the parameters at the best moments we described before (see Section 3) for the frequency (Figure 7) or the amplitude (Figure 6).

At each update time, for the concerned oscillator, the decision of switching from one synthesis method to the other can be made. And since at this update time the amplitude, frequency, and also phase of the oscillator are known, the switch of method is really straightforward.

5. PSYCHOACOUSTIC MODEL

The ReSpect library can take into account psychoacoustic phenomena (threshold of quiet and frequency masking, see [22]) in order to on-the-fly ignore inaudible partials during the synthesis process, thus saving a lot of computation time.

In the PSY module (see Figure 9), we first decide whether a partial can or cannot be detected by a listener in a noiseless environment. Then, if it turns out that this partial could be heard, we check if it is not masked by some other stronger partial.

5.1. Perceptive Scales

Human beings perceive the parameters (amplitude and frequency) of the partials on logarithmic scales.

5.1.1. Decibel Scale

The decibel (dB) scale is commonly used to represent the volume. The relation between the volume in dB and the linear amplitude is given by Equation 17:

$$V(a) = 20 \log_{10}\left(\frac{a}{A_{0\text{dB}}}\right). \tag{17}$$

If we consider that the maximal amplitude (1.0 in the linear scale) should correspond to a volume of 120 dB in order to match the standard dB SPL (Sound Pressure Level) scale, then we set $A_{0\text{dB}} = 10^{-6}$ corresponding to an acoustic pressure of $P_{0\text{dB}} = 2 \cdot 10^{-5}$ Pa (Pascals). Anyway, amplitude and pressure being proportional, it is just a matter of translation of the volume origin (0 dB) in the logarithmic scale.

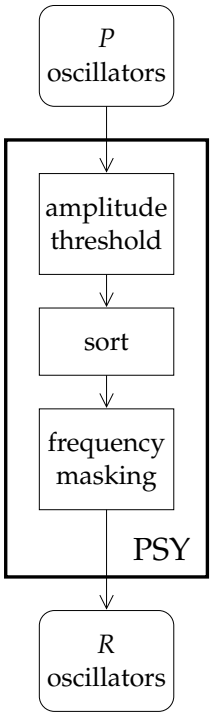

Figure 9. Diagram of the psychoacoustic (PSY) module, reducing the number of partials from P to $R \leq P$.

5.1.2. Bark Scale

A very convenient scale for representing frequencies is the Bark scale (after Barkhausen), which is very close to our perception [22]. Equation 18 allows us to go from the Hertz scale to the Bark scale:

$$B(f) = \begin{cases} f/100 & \text{if} \quad f \leq 500 \\ 9 + 4\log_2(f/1000) & \text{if} \quad f > 500 \end{cases} \tag{18}$$

5.2. Threshold of Hearing

Human beings can hear frequencies in the range of 20 Hz to 20 kHz approximatively, but the sensibility threshold in amplitude $\mathcal{S}_a$ is a function of frequency (see Figure 10). Equation 19 provides us with a good approximation for this threshold. Partials with volumes below this threshold will not be heard, and thus can safely be ignored at the synthesis stage.

$$\mathcal{S}_a(f) = \begin{cases} 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} \\ \qquad +10^{-3}(f/1000)^4 \end{cases} \tag{19}$$

5.3. Frequency Masking

Physically, the addition of two signals of the same amplitude is ruled by a nonlinear addition law and

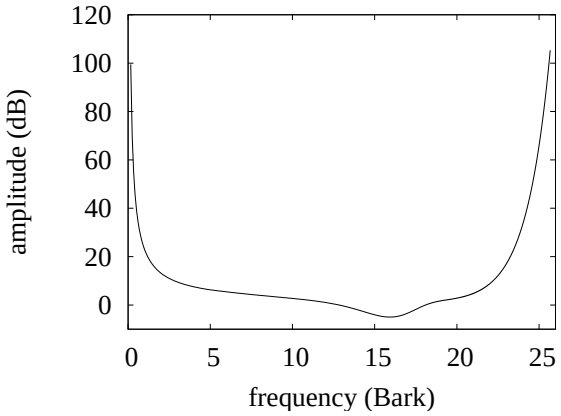

Figure 10. Threshold of hearing $\mathcal{S}_a$.

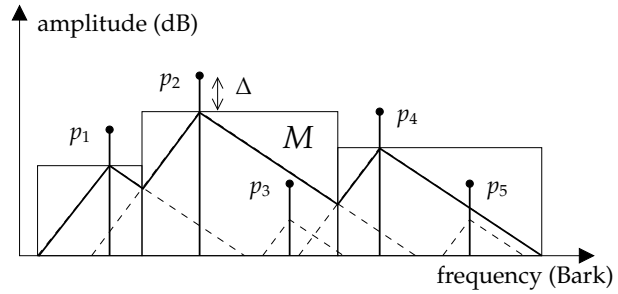

Figure 11. Five partials and the associated mask M (bold polyline). p_1, p_2, and p_4 are masking partials and contribute to M. The frequency areas of their contributions are represented by rectangles. p_5 is neither masking nor masked; p_3 is masked (by p_2).

gives a maximum of +6 dB. However, from a perceptive point of view, there is a modification of the perception threshold for a sound m (masked sound) when it is played together with a louder sound M (masking sound). This phenomenon is known as frequency masking. Consider the case of M and m being two sinusoids of frequencies f_M and f_m, and amplitudes a_M and a_m, respectively. Assume that $a_M > a_m$. If f_m is close to f_M, the sound m is masked by the sound M and thus becomes inaudible.

5.3.1. Masking Model

As a first approximation we can consider that the masking threshold is close to a triangle in the Bark-dB scales. After Garcia and Pampin [23], we use a simple masking model to evaluate the Signal-to-Mask Ratio (SMR) of each partial. This model consists of:

- The difference Δ between the level of the masker and the masking threshold (-10 dB);
- The masking curve towards lower frequencies (left slope: 27 dB/Bark);
- The masking curve towards higher frequencies (right slope: -15 dB/Bark).

5.3.2. Building the Mask

We propose in [6] an algorithm to decide whether a partial is masked or not, while computing the global mask M incrementally. First, the partials are sorted by decreasing amplitudes (to be able to decide incrementally if the current partial is audible or not). Then, for each partial p:

- If $M(f_p) + \Delta < V(a_p)$, then p is a **masking** partial and M must be updated with its contribution;
- If $M(f_p) < V(a_p) \leq M(f_p) + \Delta$, then p is neither masking nor masked;
- If $V(a_p) \leq M(f_p)$, then p is simply **masked**.

We use a list in order to store the contributions of the masking partials to the global mask M. Since this list is ordered by increasing frequencies, only the left and right neighbors are to be considered when inserting the contribution of a new masking partial. The new partial cannot mask them, since its amplitude is lower than theirs (remember that the partials have been previously sorted by decreasing amplitudes), but it can shorten the frequency area where they contribute to M.

The contributions of the masking partials to the global mask M are stored in a double-linked list, sorted by increasing frequencies. In order to decide if a new partial p is masking, neither masking nor masked, or simply masked, we need to search the list for the two contributions surrounding its frequency f_p. If it turns out that p is a masking partial, then its contribution must be inserted into the list at the right position – in order to maintain the order of the list – and the contributions of its neighbors are to be updated.

As a consequence, we need a data structure ordered in frequency, with the notion of left and right neighbors, and where searching and inserting is as fast as possible. Thus, we choose a tree-like structure, the skip list, introduced by Pugh in [24] as an alternative to balanced trees. Both data structures show a logarithmic complexity for searching.

6. SPATIALIZATION

Within ReSpect, each sound source has a spatial location (in a cartesian space). The listener has a position too, as well as an orientation (the direction towards he/she is looking at). Both the listener and the sound sources can move. Thanks to the spatialization (SPA) module, ReSpect generates sound signals incorporating this spatial information.

A sound source radiates spherical acoustic waves, that propagate to the ears through an energy transmission between air particles of the surrounding environment. For now, ReSpect considers the sound

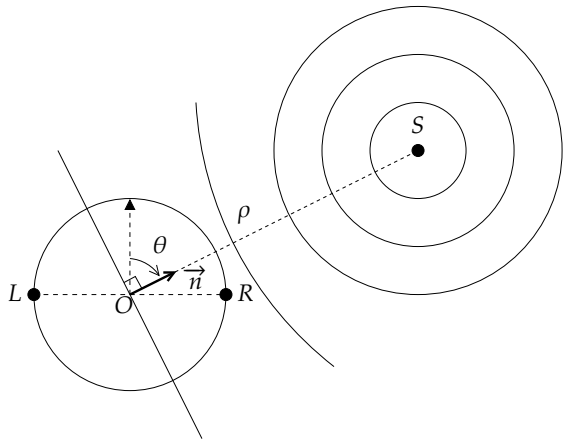

Figure 12. A source S positioned in the horizontal plane at azimuth θ, propagating acoustic waves to the head.

sources as punctual and omni-directional.

In a polar coordinate system (see Figure 12), the source point is localized given its (ρ, θ, ϕ) coordinates, where ρ is the distance between the source and the head center (O), θ is the azimuth angle, and ϕ the elevation angle.

Currently, ReSpect deals only with sources that are approximately in the same horizontal plane as the ears ($\phi = 0$). This is the case in many musical situations, where both the listener and instrumentalists are standing on the (same) ground.

Moreover, for now we consider that we are in outdoors conditions. We consider neither any room nor any obstacle (free-field case). We consider that acoustic waves are propagating in the air, at 20 degrees Celsius, 101325 Pa (Pascals) – one atmosphere, and with a relative humidity of 50%.

The listener and the sources can move. However, their speed is limited to $c \approx 343 m/s$, "the speed of sound" (Mach 1). Their acceleration is also limited, to avoid clicks. These limitations are handled by a special cinematic module within ReSpect.

6.1. Wave Propagation

6.1.1. Delay

The acoustic waves propagate in the air at a speed c. As a consequence, covering the distance ρ takes a time $\Delta_t = \rho/c$, and thus this is not the current source parameters that are heard, but the ones that were input into ReSpect Δ_t seconds before. Thus, the ReSpect library manages a buffer of parameters for each source, to be able to recover the true parameters.

6.1.2. Air Attenuation

For spherical acoustic waves, the amplitude of the source is divided by 2 (decreases by ≈ -6 dB) when measured at a distance ρ multiplied by 2.

Moreover, we also consider the frequency-selective attenuation by the air, which changes the brightness of the sound, this attenuation being roughly proportional to f^2. Since we consider that ρ is large enough for the waves to be regarded as planar when reaching the ears, the attenuation factor is given by the ISO 9613-1 norm [25].

6.2. Doppler Effect

If either the source or the listener move, the Doppler effect may affect the perceived frequencies of the partials. The Doppler factor is thus computed for each source, and applied (multiplied) to the frequencies of all of its partials. For a given source, the Doppler factor is:

$$\text{Doppler} = \frac{c + \overrightarrow{n} \cdot \overrightarrow{v}_O}{c + \overrightarrow{n} \cdot \overrightarrow{v}_S} \tag{20}$$

where $\overrightarrow{n}$ is the normalized listener-to-source orientation vector, $\overrightarrow{n} = \overrightarrow{OS}/|OS|$ (see Figure 12), $\overrightarrow{v}_O$ and $\overrightarrow{v}_S$ are the speed vectors of respectively the listener and the source, and $\cdot$ denotes the scalar product among vectors.

6.3. Spatial Cues

Although multi-diffusion with more than 2 loudspeakers is part of our current research, for now ReSpect generates only stereophonic (binaural) signals, since human beings have only 2 sensors (ears)...

The source (S) will reach the left (L) and right (R) ears through different acoustic paths, characterizable with a pair of Head-Related Transfer Functions (HRTF). A sound source positioned to the left will reach the left ear sooner than the right one, in the same manner the left level should be higher due to head-shadowing. For a given listener, the head, the torso, and the outer-ear geometry modify the sound activity content by reflections and shadowing effect.

After Viste [26], we use in [27] a simplified model, based on the main spatial (acoustic) cues for the human auditory system localization: the difference in amplitude or interaural level difference (ILD, expressed in decibels – dB) and difference in arrival time or interaural time difference (ITD, expressed in seconds).

These binaural cues can be related to physical parameters such as the celerity of sound c and the head radius $r \approx 7.5$ cm. From the analysis of the CIPIC database [28], Viste [26] derives the following models for the ILD and the ITD:

$$\text{ILD}(\theta, f) = \alpha(f)\ \sin(\theta) \tag{21}$$

$$\text{ITD}(\theta, f) = \beta(f)\, r\, (\sin(\theta) + \theta)\, /c \tag{22}$$

where α and β are frequency-dependent scaling factors that encapsulate the head / ears morphology (see Figure 13). To be listener-independent, we use the mean of individual scaling factors over the 45 subjects of the CIPIC database. For each subject, we

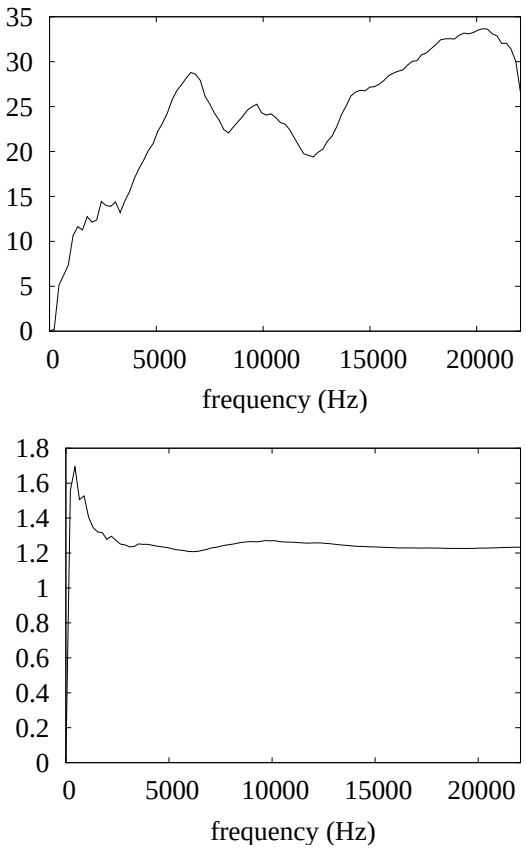

Figure 13. The α (top) and β (bottom) scaling factors. For example, β takes into account the fact that the head is not perfectly spherical.

measured the interaural cues from the HRTF and derived the individual scaling factors that best match the model – in the least-square sense – for all azimuths.

In the ReSpect library, each (monophonic) partial generates a pair of (left and right) signals, the left and right amplitudes being each modified by minus/plus the half of the ILD. The ITD is managed in the time domain. More precisely, the sound samples are written in the output buffer at a variable writing position, depending on the ITD. This imposes the use of a temporal buffer of approximatively 1.6 ms (to cover the range of all possible ITDs).

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have described the software architecture and technical details of the ReSpect library. This library is free software, under the terms for the GNU General Public License.

ReSpect encapsulates the research about spectral sound synthesis done at the SCRIME, including fast additive synthesis (SYN module), psychoascoustics (PSY module), and spatialization (SPA module).

ReSpect can be used for teaching purposes, since its source code is available, and since nearly all modules can be enabled of disabled, in order to study their effects on the synthesis speed and quality.

The ReSpect library, in its old version called libSAS, is used for example in the Dolabip project [5], a multi-field project for developing early-learning games for electro-acoustic music at school. It is also used in other computer music software tools of the SCRIME, more oriented towards composition (*e.g.* [29] and [30]).

We plan to release a new version of ReSpect by then end of 2007, and we hope that it will be useful for many researchers, teachers, and musicians. And, of course, software developers are welcome to contribute!

8. ACKNOWLEDGMENTS

First of all, I would like to thank Anthony Beurivé for his precious help with the SAS library from 2001 to 2004, while he was software engineer at the SCRIME. I'm also grateful to Robert Strandh and Matthias Robine (SYN module uses [2, 14]), Mathieu Lagrange (PSY module uses [6]), Joan Mouba (SPA module uses [27]). This research would have been impossible without Myriam Desainte-Catherine, as well as all the people of the SCRIME involved in the SAS library and the Dolabip project.

This research was carried out in the context of the SCRIME *(Studio de Création et de Recherche en Informatique et Musique Électroacoustique)* project which is funded by the DMDTS of the French Culture Ministry, the Aquitaine Regional Council, the General Council of the Gironde Department and IDDAC of the Gironde Department. The SCRIME project is the result of a cooperation convention between the *Conservatoire National de Région of Bordeaux*, ENSEIRB *(École Nationale Supérieure d'Électronique, Informatique et Radiocommunications de Bordeaux)* and the University of Bordeaux 1. It is made of composers of electroacoustic music and scientific researchers. It is managed by the LaBRI *(Laboratoire Bordelais de Recherche en Informatique)*. Its main goals are research and creation, diffusion and pedagogy.

9. REFERENCES

[1] S. Marchand and R. Strandh, "InSpect and ReSpect: Spectral Modeling, Analysis and Real-Time Synthesis Software Tools for Researchers and Composers," in *Proc. ICMC*, 1999, pp. 341–344.

[2] R. Strandh and S. Marchand, "Real-Time Generation of Sound from Parameters of Additive Synthesis," in *Proc. JIM*, 1999, pp. 83–88.

[3] M. Desainte-Catherine and S. Marchand, "Structured Additive Synthesis: Towards a Model of Sound Timbre and Electroacoustic

Music Forms," in *Proc. ICMC*, 1999, pp. 260–263.

[4] S. Marchand, "Musical Audio Effects in the SAS Model," *Journal of New Music Research*, vol. 30, no. 3, pp. 259–269, 2001.

[5] M. Desainte-Catherine, G. Kurtag, S. Marchand, C. Semal, and P. Hanna, "Playing With Sounds as Playing Video Games," *ACM Journal: Computers in Entertainment*, vol. 2, no. 2, pp. 16, 2004, (22 pages).

[6] M. Lagrange and S. Marchand, "Real-Time Additive Synthesis of Sound by Taking Advantage of Psychoacoustics," in *Proc. DAFx*, 2001, pp. 5–9.

[7] X. Serra, *A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic plus Stochastic Decomposition*, Ph.D. thesis, CCRMA, Dept. of Music, Stanford University, 1989.

[8] S. Marchand, *Modélisation informatique du son musical (analyse, transformation, synthèse) / Sound Models for Computer Music (analysis, transformation, and synthesis of musical sound)*, Ph.D. thesis, LaBRI, Université Bordeaux 1, France, 2000.

[9] J. A. Moorer, "Signal Processing Aspects of Computer Music – A Survey," *Computer Music Journal*, vol. 1, no. 1, pp. 4–37, 1977.

[10] R. J. McAulay and T. F. Quatieri, "Speech Analysis/Synthesis Based on a Sinusoidal Representation," *IEEE Trans. ASSP*, vol. 34, no. 4, pp. 744–754, 1986.

[11] X. Serra and J. O. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[12] IRCAM, Paris, *AudioSculpt User's Manual*, second edition, 1996.

[13] S. Marchand, "Compression of Sinusoidal Modeling Parameters," in *Proc. DAFx*, 2000, pp. 273–276.

[14] M. Robine, R. Strandh, and S. Marchand, "Fast Additive Sound Synthesis Using Polynomials," in *Proc. DAFx*, 2006, pp. 181–186.

[15] S. Marchand and M. Raspaud, "Enhanced Time-Stretching Using Order-2 Sinusoidal Modeling," in *Proc. DAFx*, 2004, pp. 76–82.

[16] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics – Principles and Practice*, chapter 11: Representing Curves and Surfaces, pp. 471–531, System Programming Series. Addison-Wesley, second edition, 1995.

[17] A. Freed, X. Rodet, and P. Depalle, "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware," in *Proc. ICSPAT*, 1992, pp. 98–101.

[18] A. Freed, X. Rodet, and P. Depalle, "Performance, Synthesis and Control of Additive Synthesis on a Desktop Computer Using FFT^{-1}," in *Proc. ICMC*, 1993.

[19] J. W. Gordon and J. O. Smith, "A Sine Generation Algorithm for VLSI Applications," in *Proc. ICMC*, 1985, pp. 165–168.

[20] T. Hodes and A. Freed, "Second-order Recursive Oscillators for Musical Additive Synthesis Applications on SIMD and VLIW Processors," in *Proc. ICMC*, 1999, pp. 74–77.

[21] N. Meine and H. Purnhagen, "Fast Sinusoid Synthesis for MPEG-4 HILN Parametric Audio Decoding," in *Proc. DAFx*, 2002, pp. 105–110.

[22] E. Zwicker and H. Fastl, *Psychoacoustics Facts and Models*, Springer Verlag, 1990.

[23] G. Garcia and J. Pampin, "Data Compression of Sinusoidal Modeling Parameters Based on Psychoacoustic Masking," in *Proc. ICMC*, 1999, pp. 40–43.

[24] W. Pugh, "Skip Lists: A Probabilistic Alternative to Balanced Trees," *Communications of the ACM*, pp. 668–676, 1990.

[25] International Organization for Standardization, Geneva, Switzerland, *ISO 9613-1:1993: Acoustics – Attenuation of Sound During Propagation Outdoors – Part 1: Calculation of the Absorption of Sound by the Athmosphere*, 1993.

[26] H. Viste, *Binaural Localization and Separation Techniques*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 2004.

[27] J. Mouba and S. Marchand, "A Source Localization/Separation/Respatialization System Based on Unsupervised Classification of Interaural Cues," in *Proc. DAFx*, 2006, pp. 233–238.

[28] V. R. Algazi, R. O. Duda, D. M. Thompson, and C. Avendano, "The CIPIC HRTF Database," in *Proc. IEEE WASPAA*, 2001, pp. 99–102.

[29] S. Marchand, "ProSpect : une plate-forme logicielle pour l'exploration spectrale des sons et de la musique," in *Proc. JIM*, 2000, pp. 31–40, in French.

[30] A. Beurivé, "Un logiciel de composition musicale combinant un modèle spectral, des structures hiérarchiques et des contraintes," in *Proc. JIM*, 2000, pp. 21–30, in French.