SPECTRAL ENVELOPES AND INVERSE FFT SYNTHESIS

X. Rodet[*] & P. Depalle

IRCAM, 31 rue Saint Merri,
75004, Paris, France

Abstract

We present a new additive synthesis method based on spectral envelopes and inverse Fast Fourier Transform (FFT^{-1}). User control is facilitated by the use of spectral envelopes to describe the characteristics of the short term spectrum of the sound in terms of sinusoidal and noise components. Such characteristics can be given by users or obtained automatically from natural sounds. Use of the inverse FFT reduces the computation cost by a factor on the order of 15 compared to oscillators. We propose a low cost real-time synthesizer design allowing processing of recorded and live sounds, synthesis of instruments and synthesis of speech and the singing voice.

Introduction

Many musical sound signals may be described as a combination of a pseudo-periodic waveform and of colored noise [1]. The pseudo-periodic part of the signal can be viewed as a sum of sinusoidal components, named *partials*, with time-varying frequency and amplitude [2]. Such sinusoidal components are easily observed on a spectral analysis display (Figure 1), as obtained, for instance, from a bank of filters or from a Discrete Fourier Transform.

In consequence, some of the first attempts at sound synthesis were based on the method called *additive synthesis*, that is the summation of time-varying sinusoidal components [3],[4]. This *signal modeling* approach inherits a rich history of signal processing techniques. As an example, we have developed methods to automatically analyze sounds in terms of partials and noise that can then be applied directly to additive synthesis [5]. In the sinusoidal model, harmonic or inharmonic partials are easy to characterize and easy to synthesize. Another interesting aspect of additive synthesis is the simplicity of the mapping of partial parameters (frequency and amplitude) into the human perceptual space. These parameters are meaningful and easily understood by musicians.

Thus, additive synthesis is accepted as perhaps the most powerful and flexible method. However, it appears that its development and use have been discouraged by

[*] Presently on sabbatical at Center for New Music and Audio Technologies, University of California, Berkeley

severe drawbacks. This is why we have developped a new additive synthesis method based on spectral envelopes and inverse Fast Fourier Transform that we name FFT^{-1}.

The first drawback of the classical *oscillator method* of additive synthesis is the computation cost which can easily be seen by considering a sound such as a low pitch piano note that can sometimes have more than a hundred partials. In the following we will examine this cost in detail and find a gain of 10 to 30 in favor of FFT^{-1} versus the classical method. The second drawback of the oscillator method of additive synthesis is the difficulty of introducing precisely controlled noisy components. Noise components are very important for realistic sounds and musical timbres. As an example, speech or the Japanese Shakuhachi flute cannot be created without noise. Our method makes noisy components easy to describe and cheap to compute. Last but not least, controlling hundreds of sinusoids is a great challenge for the computer musician. We will explain how a scheme based on spectral envelopes renders this control more simple, direct and user friendly.

Theory and the oscillator method

Additive synthesis is usually done with a bank of sinusoidal oscillators. Let us call J the number of partials of the signal to be computed at a certain time, that is at a certain sample n. Let the frequency, the amplitude and the phase of the jth partial, $1 \le j \le J$, be named respectively f_j, a_j, and ϕ_j. More precisely, since they are functions of time, i.e. of n, we write them $f_j[n]$, $a_j[n]$, and $\phi_j[n]$. Usually, $f_j[n]$ and $a_j[n]$ are obtained at each sample by linear interpolation of the breakpoint functions which describe the evolution of f_j and a_j. The phase, which is often ignored, requires a different procedure and we delay its treatment to a further section. The jt^{h} partial is therefore defined by:

$$c_j[n] = a_j[n].\cos(\Phi_j[n])$$

$$\Phi_j[n] = \Phi_j[n-1] + 2\pi \frac{f_j[n]}{Sr}$$

and the signal to be computed is:

$$s[n] = \sum_{j=1}^{J} c_j[n]$$

In the the oscillator method, the instantaneous frequency and amplitude are calculated first by interpolation. Then the phase $\Phi_j[n]$ is computed. A table lookup is used to obtain the sinusoidal value of this phase and the sinusoidal value is multiplied by $a_j[n]$. Finally $c_j[n]$ is added to the values of the j-1 previous partials already computed. The computation cost $C_{add}(J)$ of the oscillator method is of the form $\alpha.J$ per sample, where α is the cost of at least 5 additions, 1 table lookup, 1 modulo 2^p, and 1 multiplication. Its value is very dependant of the processor being used. Even though it is possible to modify the sinusoidal oscillator in order to produce large or narrow band limited random signals by combined amplitude and phase modulation, this has rarely be done.

Inverse Fast Fourier Transform additive synthesis (FFT^{-1})

In our method [6], the computation of the partials is not done by a bank of oscillators but by an Inverse Fast Fourier Transform, that is a transformation of a short term spectrum (STS) $S_l[k]$ into the corresponding time-domain signal $sw_l[n]$. To better explain the method, let us consider first an analysis method familiar to many people, the analysis by Fast Fourier Transform such as used in the Phase Vocoder [7]. In this analysis, the signal s[n] is first cut into successive frames $s_l[m]$ which overlap. Each frame is mutiplied by a so called *window* signal w[m] such as the well known Hanning window (Figure 2). Note that with an appropriate choice of w and of the overlapping factor d (typically d=0.5), s[n] can be exactly reconstructed from the windowed frames $sw_l[m]$ by the so called *overlap-add* method. Then the complex STS of each frame is computed by FFT, leading to a succession of spectra $S_0[k]$, $S_1[k]$, $S_2[k]$, ... Since spectra $S_l[k]$ are complex valued, as an example of a STS, on figure 1 we present its magnitude. Now the FFT^{-1} method is easily understood as just the inverse process. That is, start from the spectra $S_l[k]$, compute their Inverse Fourier Transform to get the $sw_l[m]$ and overlap-add them in order to obtain the time-domain signal s[n].

It should be clear that any signal can be exactly constructed in this manner, provided exact successive spectra $S_l[k]$ are computed and given to the inverse transform. But for reasons of efficiency we make some approximations. First, looking at figure 1, one note that a partial is represented in a spectra by a few points of non-negligible magnitude, typically K=9. This means that to build the contribution of a given partial in the STS $S_l[k]$, we only have to compute these K spectral values and add them to $S_l[k]$. If N is the size of a frame (typically N=256), we find here a gain in computation roughly proportional to N.d/K = 15. The second approximation is optional. If the parameters of a partial are slowly varying we may consider them constant on one frame i.e. consider that we have a pure sinusoid during the small interval of one frame. The advantage of this approximation is that we can then use a window w which is symmetrical around its center and thus has a real spectrum. As explained further, this replaces a complex multiplication (implemented as 4 real multiply and 2 real add instructions) by 2 real ones. However, this advantage is balanced by some drawbacks. Since the window w has to be symmetrical, fast changes in amplitude or frequency forces the use of short windows, thereby decreasing the computational saving of the method. This tradeoff can be adjusted according to the application, and the processor in use.

Construction of a STS

Let us describe exactly the construction of a STS presented in the previous section. Let f_j, a_j and ϕ_j be the mean values of $f_j[n]$, $a_j[n]$, $\phi_j[n]$ on the frame l. Let W be the Fourier Transform of the window w. Then the contribution of the partial j in the STS $S_l[k]$ is $a_j.e^{i\phi_j}.W[f_j-k]$, which means that W is shifted in order to be centered around f_j and multiplied by the complex amplitude $a_j.e^{i\phi_j}$. Naturally this complex contribution is added in $S_l[k]$to the contributions of the other partials. Note that, as announced in the previous section, here appears a complex-complex or real-complex multiply between $a_j.e^{i\phi_j}$ and $W[f_j-k]$ according to whether W is complex or real .

Optimizations

The cost in computation of the inverse FFT is a fixed cost, independent of the number of components. It amounts to only a fraction of what today's processors are able to achieve. On the contrary, the cost of the construction of a STS is proportional to the number J of components and therefore needs to be diminished. This cost decreases with the number K of non-negligible values in the spectrum W of the window. For this reason a *good* window has few non-negligible values. But for an exact construction of signals, the sum of two overlapped windows should amount exactly to 1 in the overlapped region. Moreover, in this region the amplitudes and the frequencies of the partials are *interpolated* by the windows, from the value in one frame to the value in the next frame. This second requirement suggests a triangular window which is not compatible with the first requirement. A nice solution to this problem is found when one notes that the first requirement appears in the frequency domain while the other appears in the time domain. Then two windows can be used. The first window named w_1 has as few non-negligible values as possible and is used to compute the contribution of each partial to the STS $S_l[k]$ as explained previously. Then the result of the inverse transform of $S_l[k]$ is the time-domain windowed frame $w_1[m].s[m]$ where $s[m]$ is the signal that we want to produce. Then a second window w_2 is applied. It is defined by $w_2[m]=tr[m]/w_1[m]$, where $tr[m]$ is the triangular symmetric window (figure 3). The result of this application is:

$$w_1[m].s[m].tr[m]/w_1[m] = s[m].tr[m],$$

fulfilling our second requirement of a triangular window on the time-domain signal. Note that this procedure is not costly since it is applied globally for all the partials. But it insures that the overlap-add uses frames of signal with a triangular window, $s[m].tr[m]$. As we said, the advantage is that amplitudes of partials are linearly interpolated as in the classical method. Moreover, we explain in the following section how it permits, by an adequate phase adjustment, to remove a great deal of the amplitude modulation distortion which may appear when the frequency of a partial vary rapidly from one frame to the next.

Finally, note that the number K of significant values in the spectrum W of the window can be adjusted at best by use of an auditory model. As a simple example, partials with low amplitude obviously require a smaller K.

Phase adjustment

Phase and frequency of a partial cannot be given independent arbitrary values without some care. In general, when both frequency and phase are given, these values have been estimated on a signal by some analysis method [5]. Then the phase is known with a rather good precision in a 2π interval. But the frequency is less precise and needs to be corrected in order to be compatible with the phase information. When this is done, the frequency $f(t)$ and the phase $\phi(t)$ can be defined exactly at any instant t.

To avoid possible confusion, we now use superscripts for the indexes l and $l+1$ of successive frames: let f^l, a^l and f^{l+1}, a^{l+1} be the frequency and amplitude of a partial in these frames. When $f^l \neq f^{l+1}$, some amplitude modulation can occur by phase cancellation in the overlapped region as shown in figure 4. This modulation is maximum at the point where the two triangular windowed overlapped sinusoids are of equal amplitude. The corresponding instant τ is easy to find because the window w2 is triangular hence linear. Now the modulation can largely be reduced if the phase function in each frame is chosen so as to be equal at time τ. Naturally the chosen value is the exact phase $\phi(\tau)$ determined as explained in the previous paragraph.

Noise components

Noise components of a natural sound appear in its STS as in figure 1. In our FFT^{-1} synthesis method, we can introduce noise components precisely in any frequency band narrow or wide and with any amplitude. We simply add in the STS under construction, at proper places, bands of STS's of w_1 windowed white noise or of any noise signal. This is easy and inexpensive if the STFT has been computed and stored in a table before the beginning of the synthesis stage. There exist analysis methods [5],[8],[9] to separate the noise components from the sinusoidal ones, allowing the preparation of tables of noise component STS's.

Implementation and performances

The algorithm has been implemented first in UDI [10] and in real time on the SUN-Mercury Workstation [11] and SGI Indigo [12]. In terms of cost, one of the critical part of the FFT^{-1} algorithm is the construction of a STS S. For each partial, the critical part consists of K iterations of the form: $a_j.e^{i\phi_j}.W[f_j\text{-}k]$. For efficiency, W is tabulated and hence over-sampled by a factor q (typically $q=2^8$). This table can be stored in an interleaved way such that the K successive values to read are in consecutive memory locations. Let $W_{j,k}$ such a real value read from the table. The complex value $a_j.e^{i\phi_j}$ can be computed before the loop K in form of the couple of real values (x, y). Then the instructions in the loop are:

- read $w_{j,k}$ from the next location in the table W,
- compute $x*w_{j,k}$ and $y*w_{j,k}$,
- store $x*w_{j,k}$ and $y*w_{j,k}$ in the next locations of the complex STS S.

The loop itself and the data preparation for the loop can be done with few cycles on modern processors. The number of cycles is on the order of 80 on a RISC processor. On a superscaler, this number should be divided by a factor 2. As an example on the Mips 3000 of the SGI Indigo [10], on the order of 200 partials can be computed in real time at 44.1KHz (N=256) where one third of the CPU time is used for the inverse FFT algorithm itself. Note that noise components are far less costly than partials.

In general, the computation cost is of the form $C_{fft}(J) = \beta + \gamma J$ where β and γ depend on the processor in use. Comparing $C_{fft}(J)$ and $C_{add}(J) = \alpha J$, we have the following results. When $\alpha < \gamma$, there exists a lower limit $L = \beta / (\alpha - \gamma)$ under which the oscillator method is more efficient than FFT^{-1} (typically L=20). But the saving in computation cost increases with the number of partials. There also exists an asymptotic maximum gain of computation $G = \alpha / \gamma$ (typically G=30).

Control by spectral envelopes

In usual implementations of additive synthesis, $f_j[n]$ and $a_j[n]$ are obtained at each sample by linear interpolation of breakpoint *functions of time* which describe the evolution of f_j and a_j versus time. When the number of partials is large, control by the user of each individual breakpoint function becomes impossible in practice. Another argument against such breakpoint functions is as follows. In the case of the voice and of certain instruments, a source filter model [13] is an adequate representation of some of the behavior of the partials. Then the amplitude of a component is a function of its frequency, i.e. the transfer function of the filter [14]. That is, the amplitude a_j can be obtained automatically by evaluating some spectral function (Figure 5), named *spectral envelope* $e(f_j)$, at the frequency f_j of the partial j. The amplitude variations induced by frequency variations such as vibrato can eventually be very large [14]. In consequence, a breakpoint function of time cannot neglect these amplitude variations and thus may have many breakpoints. Moreover, the amplitude of a partial is then not an intrinsic property of the sound independent of other characteristic such as fundamental frequency. Therefore the amplitude should not be stored in a breakpoint function of time, otherwise modifications of fundamental frequency or vibrato cannot be applied.

On the contrary, a spectral envelope such as the one in figure 5 can be described as an analytical function of a few parameters, whatever number of partials it is used for. It can vary with some of its parameters for effects such as spectral tilt or spectral centroid changes known to be related to loudness and brilliance. Spectral envelopes can be obtained automatically by different methods among which Linear Prediction analysis [15]. If the amplitudes and frequencies of the partials are already known from sinusoidal analysis, the Generalized Discrete Cepstral analysis [16],[17], provides reliable envelopes, the smoothness of which can be adjusted according to the order as in classical cepstral analysis. From the previous arguments, it appears why we describe amplitudes of partials by use of successive spectral envelopes defined at specific instants, for example the beginning and the end of the attack , sustain and decay of a note, etc....[18]. Then at any instant t, the spectral envelope to be used is obtained by interpolation between two successive envelopes. Let us also say that the specific instants can also be found automatically for the optimal reconstruction of the sound by linear interpolation between two successive envelopes [19]. No need to add that an amplitude spectral envelope is even better adapted for the description of noise components.

Frequencies and phases also can be described by similar envelopes that we call *generalized spectral envelopes*. As an example, sustained sounds have partials with frequencies very close to harmonic values of the fundamental frequency. It is therefore advantageous to record only the deviation from harmonicity as a function, called *frequency deviation envelope*, of the harmonic number or of the harmonic frequency. Then, at the synthesis stage, it is easy to compute the frequency of a partial, but we still have the control of the amount of inharmonicity in the resulting sound. The same procedure can be applied for the phases of the partials.

Conclusion

Our new method of additive synthesis by FFT^{-1} brings a solution to the three main difficulties of classical additive synthesis. The processing time (calculation cost) can be divided by a factor of up to 20. Apart from one inverse FFT - the cost of which is independent of the number of voices and of sinusoid and noise components - the calculation is reduced to the *construction* of the Short Term Spectrum. By optimization in time and frequency domains, we have found that this construction can be limited, for each sinusoid, to simple table lookups and multiplications of a small number of values, such as 9. This can be compared with the possibly 128 values in the short term window to be computed with the usual method. As in the case of oscillator-based additive synthesis, our procedure can build sinusoids with rapidly varying amplitude and frequency, and with phase control if required. For each sample, the amplitude and the frequency are the values interpolated between values given at each frame corresponding to 1ms for example.

It is easy and not costly to introduce noise precisely in any frequency band narrow or wide, and with any amplitude. We just have to add into the STS under construction, at the proper places, bands of STS's of white noise or of any noisy signal extracted from natural sounds for instance. We have developed methods to automatically analyze sounds in terms of partials and noise that can then be applied directly to FFT^{-1}. Processing of sounds is therefore possible with a high degree of precision and synthesis can be done simultaneously.

In our method, control is made easier by use of *spectral envelopes* instead of the time-functions classically used for additive synthesis. Precisely, the characteristics of the sound are given in terms of features such as amplitude envelope of partials, fundamental frequency, deviation from harmonicity, noise components, etc.... These spectral envelopes need only to be defined at some specific times such as beginning of attack, end of attack, end of sustain, etc.... Then at any time, the synthesis algorithm uses envelopes interpolated between the defined envelopes. This procedure renders user control much more economic and efficient.

Finally, we also suggest that a limited cost real time multi-timbral instrument can easily be designed based on FFT-1. This would have all the possibilities of present day synthesizers, plus many others such as the precise modifications of sampled sounds, speech and singing voice synthesis. Sound examples of pure synthesis and processing are very successful.

Acknowledgments

The authors gratefully acknowledge Adrian Freed for the SGI Indigo implementation in HTM[12].

References

1. X. Rodet, "Analysis and Synthesis Models for Musical Applications", IEEE Workshop on application of digital signal processing to audio and acoustics, Oct. 1989, New Paltz, New-York, USA.

2. J. Laroche and X. Rodet, "A new Analysis/Synthesis system of musical signals using Prony's method", Proc. ICMC, Ohio, Nov. 89.

3. J.C. Risset and M.V. Mathews, "Analysis of musical-instrument tones", Physics Today, 22(2):23-30, Feb. 1969.

4. G.J. Sandell and W.L. Martens, "Prototyping and Interpolation of Multiple Musical Timbres Using Principal Component-Based Synthesis" , to appear in: Proc. Int. Comp. Music. Conf., San José, CA, USA, Oct. 1992

5. G. Garcia, "Analyse des signaux sonores en termes de partiels et de bruit. Extraction automatique des trajets fréquentiels par des Modèles de Markov Cachés" Mémoire de DEA en Automatique et Traitement du Signal, Orsay, 1992

6. P. Depalle & X. Rodet, "Synthèse additive par FTT inverse", Rapport Interne IRCAM, Paris 1990.

7. Ph. Depalle and G. Poirot, "A modular system for analysis, processing and synthesis of sound signals", Proc. of the Int. Comp. Music Conf., Montreal, Canada, 1991.

8. R.J. Mc Aulay and Th. F. Quartieri, "Speech analysis/synthesis based on a sinusoidal representation", IEEE Trans. on Acoust., Speech and Signal Proc., vol ASSP-34, pp. 744-754, 1986.

9. X. Serra, "A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition", PhD dissertation, Stanford Univ., 1987.

10. P. Depalle & X. Rodet, "UDI, a Unified DSP Interface for Sound Signals Analysis and Synthesis", Proc. Comp. Music Conf., 1990, Computer Music Association.

11. G. Eckel, X. Rodet and Y. Potard, "A SUN-Mercury Workstation", ICMC, Urbana, Champaign, USA, August 1987.

12. A. Freed, "Tools for Rapid Prototyping of Music Sound Synthesis Algorithms and Control Strategies" , to appear in: Proc. Int. Comp. Music. Conf., San José, CA, USA, Oct. 1992

13. X. Rodet, P. Depalle & G. Poirot, "Speech Analysis and Synthesis Methods Based on Spectral Envelopes and Voiced/Unvoiced Functions", European Conference on Speech Tech., Edinburgh, U.K., Sept. 87.

14. X. Rodet, Y. Potard, J.B.B. Barrière, "The CHANT Project", Computer Music Journal, MIT Press, fall 85.

15. X. Rodet, P. Depalle, "Use of LPC Spectral Estimation for Analysis, Processing and Synthesis", 1986 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New-Paltz, New York, September 1986

16 T. Galas & X. Rodet, "An Improved Cepstral Method for Deconvolution of Source-Filter Systems with Discrete Spectra", Int. Computer Music Conf., Glasgow, U.K., Sept. 90.

17. T. Galas & X. Rodet, "Generalised Discrete Cepstral Estimation of Sound Signals" IEEE Workshop on Application of Signal Processing to Audio and Acoustics, New Platz, New York, Oct. 1991.

18. X. Rodet, P. Depalle & G. Poirot, "Diphone Sound Synthesis", Int. Computer Music Conf., Koeln, RFA, Sept 88.

19. C. Montacié, "Décodage Acoustico-Phonétique: apport de la décomposition Temporelle Généralisée et de transformations spectrales non-linéaires", Thèse de l'URA 820, Telecom-Paris, Paris 1991

Figure 1. A Short Term Spectrum magnitude showing partials and noise.

Figure 2. Successive windowed frames of the signal s[n].

Figure 3. A triangular window applied to a sinusoid.

Figure 4. Overlapping frames of sinusoids with different frequencies showing possible phase cancellation.

Figure 5. Amplitude spectral envelope of a Short Term Spectrum.

τ

l-1 l l+1