A Comprehensive Study of Analysis and Synthesis of Tones by Spectral Interpolation

Marie-Helene Serra, Dean Rubine, Roger Dannenberg
June 1988
CMU-CS-88-146

Computer Science Department
Carnegie Mellon
Pittsburgh, PA 15213

Abstract

This paper presents a technique for the analysis and digital resynthesis of instrumental sounds. The technique is based on a model which uses interpolation of amplitude spectra to reproduce short-time spectral variations. The main focus of our work is the analysis algorithm—starting from a digital recording we are able to automatically compute the parameters of our model. The parameters themselves, harmonic amplitudes at selected times, are small in number and intuitively interpretable. The model leads to a synthesis technique more efficient than classical additive synthesis; moreover it allows dynamic spectral variations to be controlled with only a few high-level parameters in real time.

We have studied two analysis/synthesis methods based on spectral interpolation. The first uses only spectral interpolation. This method has allowed us to compress recordings of orchestral instruments to an average of 400 bytes per second without perceptible loss of realism, and to resynthesize these sounds with about 10 arithmetic operations per sample. The second method is a hybrid in which a sampled attack is spliced onto a sustain synthesized via spectral interpolation.

The spectral interpolation model has been applied successfully to different instruments belonging to the brass and woodwind family. We plan to extend the study to many more instruments.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under contract F33615-87-C-1499 and monitored by the: Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Aeronautical Systems Division (AFSC), Wright-Patterson AFB, Ohio, 45433-6543.

Further support for this work was provided by grants from Apple Computer, Inc., and the Yamaha Corporation, an IBM Fellowship Supplement, and a Ben Franklin Partnership Fund Matching Award, Agreement Number 402995-45176-308-001.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies, the U.S. Government, or Carnegie Mellon.

Table of Contents

List of Figures

1. Introduction

This paper explores techniques for computer analysis and synthesis of musical sounds based on the interpolation of spectra. In section 2, after having examined the technical and musical performance of some existing synthesis algorithms, we list the goals of our work. In section 3 we introduce the method of waveform interpolation synthesis (of which spectral interpolation synthesis is a special case), then we describe the architecture and control of a waveform interpolation oscillator. Section 4 argues the need for an automatic and general analysis model based on interpolation. Here we discuss the conditions a sound must satisfy in order to be amenable to synthesis via waveform interpolation, and introduce the technique of analysis/synthesis by spectral interpolation. Section 5 describes each step of the analysis process using spectral interpolation: digital recording, spectral analysis, and data reduction. We propose two different data reduction algorithms based on "linear" spectral interpolation (meaning linear in time), and another algorithm based on "nonlinear" spectral interpolation. Then we analyze and compare the results obtained with these two algorithms on several types of instruments. This evaluations leads to an improved technique which combines sampling and spectral interpolation synthesis. This hybrid technique forces us to confront the problem of achieving an inaudible transition between sampled and synthesized sound. We discuss two approaches for solving this problem—one successful, the other not. Section 6 situates our research in relation to other relevant work. Finally we outline directions for future work.

2. Evaluation of the Existing Digital Synthesis Techniques

In this section we list some criteria by which we can assess synthesis algorithms; we then evaluate many current synthesis techniques according to these criteria. Given this groundwork, we argue that there are desirable points in this space of criteria which are currently achievable, but which until now have not been explored. This leads to the synthesis technique of waveform interpolation, which we describe and evaluate in subsequent sections of this paper. The reader should feel free to skip to section 3 if he wishes to learn about the technique before learning about its underlying motivation.

2.1. The Criteria

Each digital sound synthesis algorithm may be viewed as a model of sound production; the model takes as input a number of parameters which collectively determine the sound produced. The production model is usually represented by a mathematical formula whose variables are the input parameters and whose result is the sequence of output samples. Most of the sound generation systems let the user manipulate some of the input parameters to obtain different output audio signals.

Any synthesis algorithm may be judged according to how effectively it enables the musician to produce musically useful sounds. Effectiveness can be measured in terms of specific criteria such as the *generality* of the synthesis method, the computational *efficiency* of the corresponding algorithm, and the quality of the *control* over the sound available to the user. In what follows, we propose a definition of each of these criteria; we also explain why it is important that a digital sound system includes features that tend to maximize these criteria.

generality

Generality is the ability to generate many kinds of sounds. We can distinguish between two classes of digital sounds. The first is the class of the so-called "natural" or "natural-like" sounds which attempt to imitate the signals produced by acoustical systems such as orchestral instruments and the human voice. The other class of digital sounds, which we call "non-natural" are not attempts to sound like an acoustic system. The generality of a synthesis algorithm is determined by the variety of

sounds, both natural and not, which it is able to produce.

It is clearly desirable for a digital sound system to be general, so that it enables the composer to create a wide range of sonorities, resulting in music whose orchestration is rich and creative. The combination of electronic and physical worlds that a general system could provide may be musically fruitful [52]. The ability to digitally simulate many traditional instruments is good in its own right, since it allows a composer to have a full traditional orchestra always at his disposal. Furthermore, a general system that allows the modification and combination of digital representations of existing instruments to create new ones has interesting musical applications [5].

efficiency Efficiency refers to the the computational cost of the synthesis algorithm measured in computational operations per sample of synthesized sound. Greater efficiency implies some combination of reduced overall hardware cost, greater number of synthesized "instruments", and in non-real-time systems, shorter computation time. All of these are practical advantages.

control By controlling the input parameters of the synthesis algorithm the user attempts to achieve a desired perceptual result. Depending on the nature of the input parameters, and on the nature of the algorithm itself, the synthesis system is more or less usable. We think that the following aspects of control are important: *timbral control*, *simplicity* of control, *intuitiveness* of control, and *flexibility* of control. Control over timbre is as desirable as pitch, loudness, and duration control. Indeed, as the musical properties of an instrument come from its characteristic variations of timbre, it is of primary interest to be able to reproduce such variations and to derive new sounds from them. Furthermore, timbre has become a main concern for computer musicians. Given the potential of digital instruments, it is now possible to write and produce dynamic timbral structures as part of a musical composition [53, 54]. It is desirable that control be simple, meaning that there are only a small number of parameters to drive. Control should also be intuitive, meaning that the user should be able to easily imagine the perceptual consequences of parameter modifications. Also, a flexible synthesis model is of interest for the musician because by modifying just a few parameters he can obtain a wide range of sonorities. Simplicity, intuitiveness, and flexibility are necessary to make the sound editing task attractive.

2.2. Evaluation of Synthesis Techniques

Currently no technique simultaneously achieves all the desirata mentioned above; existing techniques emphasize some of them to the detriment of the others.

- For example FM [12] is extremely efficient, flexible, simple to control and fairly general. By varying the modulation index and the frequency ratios between carrier and modulated waves, it is possible to get a rich ensemble of natural sounds, harmonic and inharmonic, and create new timbres as well. FM can also mimic the evolution of natural spectra [31, 43]. However, some problems remain in the design of a direct, automatic and general analysis algorithm which can derive FM synthesis parameters from instrumental sounds [34]. In addition, FM parameters are not intuitive.

- With additive synthesis it is possible to reproduce exactly any kind of natural sound (harmonic and inharmonic). The input parameters of the synthesis model (amplitude and frequency time functions of each partial) can be obtained by submitting the sound to a short-time Fourier based analysis: the heterodyne filter [31, 40] gives very good results for quasi-harmonic sounds, and phase vocoder analysis [32, 17] has been used extensively for characterizing and reproducting sounds with complex harmonic structures [45]. Because the input parameters give a very accurate definition of the time-varying spectrum, additive synthesis offers a powerful expressive control over the sound.[1] Furthermore, the parameters

[1] For example, interesting musical effects can be obtained by scaling the time-functions describing the evolution of the partials.

of the analysis and synthesis models lend themselves to a perceptual interpretation. Because of the properties of the representation (accuracy and intuitiveness) analysis-based additive synthesis has proved to be a very useful tool for carrying out physical and perceptual descriptions of acoustic instruments [21, 22] and fundamental studies on timbre characterization and timbre manipulation [19, 53]. Thus we see that additive synthesis is general and offers many musical possibilities. However, additive synthesis requires a large amount of data per voice (frequencies and amplitudes of every partial). This is the cause of several disadvantages: control is complex, signal production is expensive (requiring many operations per output sample), and sound representation is somewhat redundant, emphasizing local detail rather than global properties. To reduce cost, complexity, and redundancy, specific data reduction schemes can be applied (see section 3.1).

- Some early digital instruments generated sounds using fixed waveform synthesis [10] (see section 3.3). In contrast to additive synthesis, this technique uses much less data (one constant waveform, one frequency envelope and one amplitude envelope) making it simple and efficient. However it is neither flexible nor general. It can only generate quasi-periodic and harmonic sounds, and it does not provide any control over spectral modulations. For these reasons, fixed waveform synthesis is currently of minimal musical interest.

- Nonlinear techniques, such as waveshaping [39, 4], are economical, flexible, and simple to drive. They can generate and modify natural sounds. In waveshaping synthesis, the reproduction of instrumental sounds can be made by tuning the different modules (sinusoidal generator, nonlinear processors, filters, envelope generators) on a set of analyzed characteristics of the sounds [6]. Adjusting the modules is usually carried out by a trial-and-error phase that is laborious and does not always assure a perfect matching between the original and synthesized signal. For most nonlinear methods generality is limited and control is not intuitive.

- Subtractive synthesis models all consist of the output of a (usually harmonically rich) sound source passed through a parameterized filter. We distinguish between analysis-based and non-analysis-based subtractive synthesis. In analysis-based subtractive synthesis [25] a source function (usually white noise or train of periodic pulses) is fed into a time-varying digital filter. The coefficients of the filter may be obtained by linear predictive analysis of an input signal. In order to accurately reproduce instrument tones, high order filters whose coefficients change quite often are needed [31]. Evaluating analysis-based subtractive synthesis on our criteria, we see that its scores about the same as additive synthesis. The model is general, and intuitive and flexible to control. However, the large filters and high bandwidth needed make it inefficient and make control rather complex.

- Non-analysis-based subtractive synthesis has most often been employed in analog synthesizers. The filters employed in such systems are usually very simple, usually having at most three parameters. This simplicity limits the variety of sounds that can be produced. This results in an evaluation similar, though not identical, to that of FM. Non-analysis-based subtractive synthesis are simple, flexible, and intuitive to control, quite efficient, but not very general.

- In sampling machines [26, 29, 9] the output signals are obtained by playing back prerecorded digital sounds. With this technique it is possible in theory to reproduce virtually any sound, and as many as desired given enough memory. Sampling systems may include different control facilities among which are pitch transposition, filtering, amplitude envelopes, frequency envelopes, and looping of waveform segments. This combination of functions provides a large variety of means for modifying the library of recorded signals. However the design of the sounds, especially simulations of acoustic instruments, requires much effort from the user, the amount of data (the samples) is very large, and timbral control is

restricted.[2] Therefore, sampling is general and extremely efficient (there is little or no computation) but its control is not simple and lacks flexibility.

- Physical modeling synthesis directly simulates the propagation of acoustical waves inside an instrument. The instrument is usually described by a nonlinear source function exciting a linear resonator that may include acoustical losses. The waveguide filter approach maps the instrument into a set of digital filters or delay lines [48, 49], and the mechanical approach considers a set of equations expressing the movements of a system of masses and springs that models the instrument [1, 2]. Ultimately this kind of synthesis should lead to an accurate reproduction of any instrumental sound. However, the complexity of the analysis and the large amount of computation for the synthesis often make this kind of model very expensive. Furthermore, the control of physical models is not straightforward, requiring detailed knowledge of the driving forces applied by the musician to his instrument.

This succinct study about the properties of the principal digital synthesis techniques shows that there is a tradeoff between efficiency and generality (except for sampling) and that no technique scores highly on all criteria.

2.3. Our Goals

Our primary goal is to find a simple, efficient, flexible, and intuitive method of reproducing a wide range of traditional instruments. Among the techniques previously cited, only sampling, physical modeling, and additive synthesis lead to an accurate reproduction of natural sounds. We rule out sampling due to its inflexibility of control, and rule out physical modeling as it currently requires excessive computation and complex control. Additive synthesis is not entirely satisfactory as it is inefficient and requires a large amount of data. We desire a technique that keeps the advantages of additive synthesis, and simultaneously is more efficient and offers simpler control. We are willing to trade generality for efficiency and simplicity.

To achieve close similarity between natural sounds and their computed versions, we want a computed analysis model to drive the synthesis. In addition, analysis in the spectral domain has the potential advantage of timbral control and of intuitive control. In the next section we introduce an analysis/synthesis approach that has been designed to fit our goal. This approach has been inspired by both additive and table lookup synthesis.

3. Synthesis by Waveform Interpolation

This section introduces the idea of waveform interpolation as a low-cost synthesis method able to reproduce spectral variation. We first consider existing techniques for improving additive synthesis, and we see that while it is possible to reduce the bandwidth of the control data without sacrificing sound quality, synthesis is still expensive and control still complex. We thus abandon the idea of having one oscillator per partial, and take the fixed table-lookup oscillator as our starting point. We then proceed to derive an implementation of a waveform interpolating oscillator, and to demonstrate how spectral variation may be achieved using such an oscillator.

---

[2]Systems which enable velocity switching or velocity crossfading (velocity based selection or mixing of multiple samples functions, respectively) allow velocity-driven spectral variation, but the resulting effects remain insufficient if the user wants to reproduce the timbral inflections that occur with loudness in acoustic instruments. Also, timbral modifications are limited by the amount of memory.

3.1. Additive Synthesis Considered

Roughly speaking, the cost of a digital synthesis method is proportional to the number of independent oscillators and to the bandwidth of control information needed to reproduce a single voice. Many researchers have reduced the control information bandwidth using a number of different data reduction schemes. Grey [21] and Moorer [31], have proposed the use of piecewise-linear functions to approximate the amplitude and frequency functions for each partial. This method has been extensively used in digital synthesizers where the number of oscillators is large [50, 7]. Strawn [51] and Schindler [42] have extended the piecewise-linear approximation by organizing the envelopes breakpoints into hierarchical descriptions. Smith and Serra [49] use a tracking phase-vocoder analysis that retains only the essential features exhibited by the short-time spectrum. Charbonneau [11] and Sasaki [41] have sought data reduction methods that lead to a succinct and global representation of the spectral envelope.

Even when using one of the above data-reduction techniques, the cost of additive synthesis remains high. Additive synthesis requires one oscillator per partial, thus many oscillators are often necessary to synthesize a single voice. Similarly the number of parameters to control remains considerable. Therefore, to improve the efficiency of this technique, we have chosen to look further into ways of reducing the number of oscillators per voice.

3.2. An Attempt at Spectral Variation via Table Lookup

Fixed-waveform synthesis (or table-lookup synthesis) uses only one oscillator per voice. The oscillator scans a constant waveform table periodically, scaling the output by an amplitude envelope (see section 3.3). Thus, over a given time interval, only harmonic sounds with a constant spectrum can be generated. If the waveform currently addressed is instantaneously changed to a different one, the resulting spectrum will change, and by generalizing the same procedure to a set of waveforms, dynamic spectral variation can be produced. For example, smooth timbral transitions can be obtained by reading successively a sequence of waveforms that are "very close" to each other [15]. Unfortunately, avoiding perceptual discontinuities (clicks) at the switching points requires a large amount of input data [8, 44]. In other words, the changes between successive spectra need to be slight in order to be imperceptible, necessitating a large control bandwidth.

Thus, it is not very practical to use a single table-lookup oscillator to generate spectral variation, since it is costly to change smoothly between spectra. However, if we allow ourselves two table-lookup oscillators per voice we find that we can indeed change smoothly between wavetables. Using this insight, we can construct a wavetable interpolation oscillator based on two table lookups. It will be possible to drive such an oscillator by a low bandwidth control data computed by a specialized spectral analysis. Starting from the table-lookup synthesis algorithm, we now proceed to derive the wavetable interpolation synthesis algorithm.

3.3. Table-Lookup Synthesis

A periodic tone with purely harmonic partials and a slowly varying amplitude envelope can be efficiently generated with the table lookup synthesis technique [27]. The technique consists of reading repeatedly a sequence of numbers stored in a memory. The numbers represent the samples of one cycle of a digitized waveform and the memory is called a wavetable.

Table-lookup synthesis may be described by the following set of equations:

$$PhaseInc(n) = \frac{M \times Freq(n)}{srate} \tag{2}$$

$$Phase(n) = Phase(n-1) + PhaseInc(n) \quad [\bmod M] \tag{3}$$

$$y(n) = Amp(n) \times W[Phase(n)]$$

where
n is the number (index) of the sample being computed,
M is the (constant) number of samples in wavetable W,
$srate$ is the (constant) sampling rate of the output signal,
$Freq(n)$ is the instantaneous frequency at sample n,
$Amp(n)$ is the amplitude scale factor at sample n,
$W[m]$ is the m^{th} sample of the fixed wavetable W,
and
$PhaseInc(n)$ is the phase increment at sample n,
$Phase(n)$ is the phase accumulator at sample n, and
$y(n)$ is the output signal at sample n.

Figure (Figure 3-1) shows a diagram of an oscillator based on these equations.

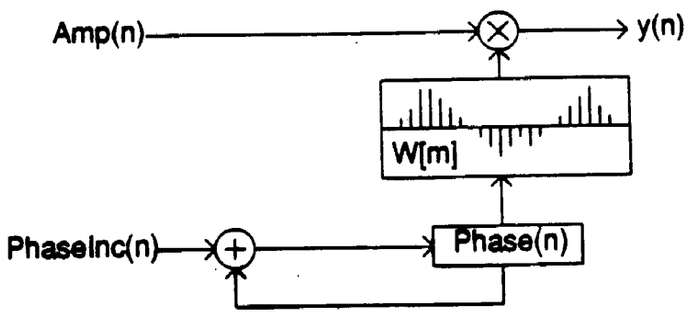

Figure 3-1: Table-lookup oscillator

It is quite straightforward to use table-lookup synthesis to create periodic tones. Setting $Amp(n)$ and $Freq(n)$ to constant functions insures periodicity (since $PhaseInc(n)$ is constant, $Phase(n)$, and thus $y(n)$ will be periodic). Furthermore pitch can be varied by varying $Freq(n)$, while an amplitude envelope may be imposed on the output by varying $Amp(n)$.

In practice, $Amp(n)$ and $Freq(n)$ are often specified as piecewise linear functions. In implementations containing ramp hardware, $Amp(n)$ and $Freq(n)$ may be updated on every sample (*i.e.* $Amp(n) = Amp(n-1) + AmpPhaseIncr(n)$). It is often the case that $Amp(n)$ and $Freq(n)$ are updated at a lower rate (*i.e.* $Amp(n) = Amp(n-1)$ except at update times). This update rate imposes an upper limit on how fast the amplitude and frequency of the oscillator is able to vary.

It is worth noting in passing that if the control functions $Amp(n)$ and $Freq(n)$ are allowed to be functions of the output of another table-lookup oscillator, complex waveforms will be generated via amplitude and frequency modulation, respectively. Such modifications will not be considered here, and from now on we shall assume that $Amp(n)$ and $Freq(n)$ are slowly varying. Given these assumptions, since the table holds a single cycle of the waveform, table-lookup synthesis can generate harmonic signals with the ratios

between harmonic amplitudes always fixed.

3.4. The Waveform Interpolation Oscillator

Waveform interpolation is similar in implementation to table lookup (figure 3-1) but uses two phase-locked wavetables, W_L and W_R, each one loaded with a different waveform (figure 3-2).

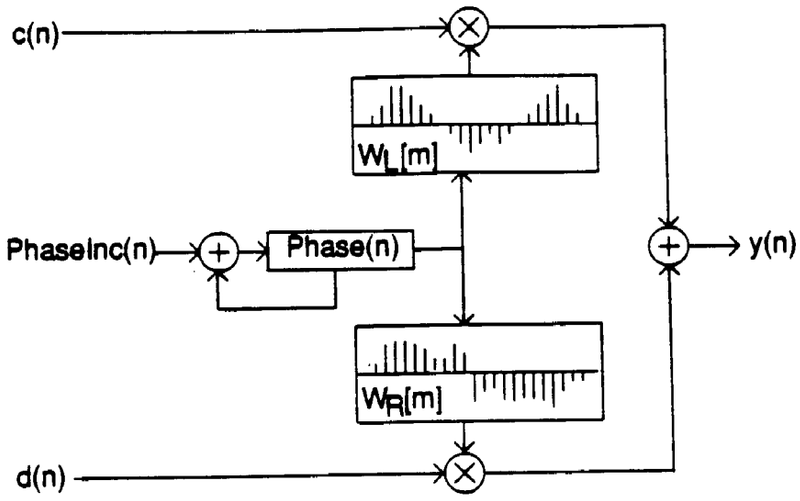

Figure 3-2: The Waveform Interpolation Oscillator

The two tables have the same length M, and are indexed with the same phase value $Phase(n)$. The interpolation signal can be expressed as

$$y(n) = c(n)\, W_L[Phase(n)] + d(n)\, W_R[Phase(n)] \tag{4}$$

where
$c(n)$ is the amplitude scale factor of the left wavetable at sample n,
$d(n)$ is that of the right,
and
n, $y(n)$, $Phase(n)$, and $PhaseInc(n)$ are as before.

Thus, we see that a waveform interpolation oscillator is identical in effect to two table-lookup oscillators whose phases $Phase(n)$ are constrained to be equal. An alternate view is to consider a waveform interpolation oscillator to be a table-lookup oscillator whose table is dynamically computed as the linear combination of two fixed wavetables.

Let us consider this latter view in a bit more detail. We rewrite equation (4) as

$$y(n) = W^E_{(n)}[Phase(n)]$$

where $W^E_{(n)}$ is the "effective" wavetable in use at sample n:

$$W^E_{(n)}[m] = c(n)\, W_L[m] + d(n)\, W_R[m] \tag{5}$$

Are we justified in calling $W^E_{(n)}$ the effective wavetable at sample n? In other words, are the short-time spectral properties of the output at sample n close to those that would have resulted had we done simple table-lookup synthesis using fixed wavetable $W^E_{(n)}$? To answer, we consider the short-time frequency domain behavior of the waveform interpolation oscillator at sample n. We expect that the output spectrum is a linear combination of the spectra of the two wavetables W_L and W_R, the combination coefficients being $c(n)$ and $d(n)$. In fact, if one assumes that $c(n)$ and $d(n)$ are constant over the window duration, then it is easy to show that the short-time Fourier transform [37] at sample n of the sequence $y(n)$ (call it $Y_n(e^{j\omega})$) satisfies

$$Y_n(e^{j\omega}) = c(n)L_n(e^{j\omega}) + d(n)R_n(e^{j\omega}) \tag{6}$$

where $L_n(e^{j\omega})$ and $R_n(e^{j\omega})$ are the short-time Fourier transforms of $W_L[Phase(n)]$ and $W_R[Phase(n)]$, respectively. Since $c(n)$ and $d(n)$ are not in fact constant, but are assumed to be slowly varying, equation (6) is only approximately true. To the extent that the spectrum $Y_n(e^{j\omega})$ is the desired linear combination of the spectra of the left and right wavetables, we are justified in calling $W^E_{(n)}$ the effective wavetable at sample n.

We see that the effect of interpolating waveforms is to interpolate the corresponding spectra. Thus, we propose to generate spectral variation through waveform interpolation.

3.5. Arbitrary Spectral Evolution via Waveforms Interpolation

We would like to use waveform interpolation (equation (4)) to generate a spectral evolution. By "spectral evolution" we mean a sound in which the ratios between the amplitudes of the harmonics change over time. There are two ways of achieving spectral evolution. First, as $c(n)$ and $d(n)$ may vary dynamically over time, the interpolated waveform, and thus the resultant spectrum, varies over time. As long as $c(n)$ and $d(n)$ are continuous the output of the oscillator will be free from clicks (being digital signals, we may interpret "continuous" as meaning "varying sufficiently slowly").

The second way of achieving spectral evolution is to switch the wavetables themselves over the course of a sound. For instance, by repeating the interpolation procedure (equation (4)) with different pairs of waveforms loaded in the right and left tables, one can get a succession of different dynamic spectral combinations. To avoid discontinuities at the point when a waveform is changed (the problem discussed in section 3.2), only one of the two waveforms is changed at any one time, and the change occurs when the scaling factor associated to the wavetable being changed is zero.

Figure 3-3 illustrates the interpolation of a succession of Q waveforms ($Q=4$) taken in a time-ordered sequence $\{ (n_0,W_0), (n_1,W_1), \ldots, (n_{Q-1},W_{Q-1}) \}$, where n_i represents the sample at which the reading of waveform W_i starts. At any point in time, we are interpolating between one of the waveform pairs $(W_0,W_1), (W_1,W_2), \ldots, (W_{Q-2},W_{Q-1})$.

In the figure we have shown $c(n)$ and $d(n)$ as piecewise-linear functions each of which alternately has value zero at some n_i, rising linearly to its maximum at n_{i+1} and again reaching zero at n_{i+2}. It is not necessary that $c(n)$ and $d(n)$ have this form; all that is required is that they be continuous and have value zero at the points their respective wavetables change.

To describe the waveform switching mathematically, we define two functions $L(n)$ and $R(n)$. Given the

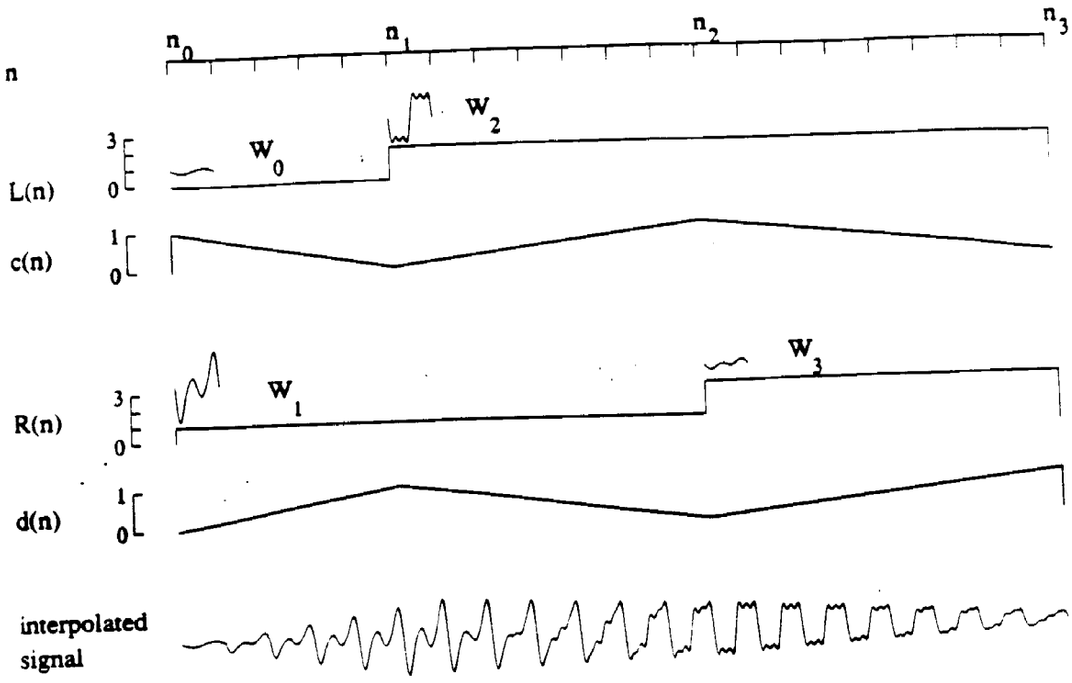

Figure 3-3: Interpolation of successive waveforms

sequence of waveforms ($W_0, \ldots, W_{Q-1}$), $L(n)$ and $R(n)$ respectively determine the waveform in the left and right wavetable of the interpolating oscillator. In a hardware implementation, these functions are used to trigger the change of waveform in each table.

We have

$$y(n) = c(n)\, W_{L(n)}[Phase(n)] + d(n)\, W_{R(n)}[Phase(n)] \tag{7}$$

For convenience, we set $n_{-1} = n_0$ and $n_Q = n_{Q-1}$. The left table will be filled with even numbered waveforms, and the right with odd ones as follows:

$$\begin{aligned} L(n) &= 2i \quad \text{where} \quad n_{2i} \le n < n_{2i+2} \\ R(n) &= 2i+1 \quad \text{where} \quad n_{2i-1} \le n < n_{2i+1} \end{aligned} \tag{8}$$

The necessary criteria to avoid clicks when switching waveforms are

$$\begin{aligned} c(n) &= 0 \quad \text{when} \quad n = n_{2i} \\ d(n) &= 0 \quad \text{when} \quad n = n_{2i+1} \end{aligned} \tag{9}$$

A particular and simple case of equation (7) is the *time-linear interpolation* of two waveforms. This is obtained by setting the two mixing coefficients $c(n)$ and $d(n)$ to two opposite linear ramps whose sum at each sample equals unity. We can rewrite this special case of equation (7) as

$$y(n) = (1-sp(n))\, W_{L(n)}[Phase(n)] + sp(n)\, W_{R(n)}[Phase(n)] \tag{10}$$

$$sp(n) = \begin{cases} (n-n_{2i})/(n_{2i+1}-n_{2i}) & n_{2i} \le n < n_{2i+1} \\ (n_{2i}-n)/(n_{2i}-n_{2i-1}) & n_{2i-1} \le n < n_{2i} \end{cases}$$

3.6. Spectral Interpolation Analysis

Problems may occur when interpolating between two waveforms whose phase distributions are different. Phase cancellation and phase shifting occur when interpolating between two waveforms whose corresponding harmonics are out of phase. Phase cancellation causes the amplitude of each harmonic in the interpolated signal to be less than expected [10]. Even when the amplitudes are the same, interpolating between two out of phase harmonics results in an unintentional but perceptible timbre change, often perceived as a frequency shift. We feel that if the phases are arbitrary, waveform interpolation often gives undesirable results.

To avoid these problems and to get an intuitive control over the interpolation process, we will only interpolate between spectra whose corresponding harmonics are all in phase. For natural sounds, it is in general not the case that the corresponding harmonics of each period are in phase. In order to conduct our experiments on waveform interpolation synthesis, we will rely on an assumption that has been extensively used in digital signal coding of speech and musical signals. The assumption is that the ear is not very sensitive to phase information, so this information can be thrown away [18]. In addition to ignoring phase shifts of given harmonic in time, we ignore the initial phase differences between different harmonics.[3] We use the term synthesis by *spectral interpolation* to mean that we have constrained the corresponding harmonics of each generator wavetable to be in phase.

Expressing the left and right wavetables of an interpolating oscillator as the sum of their respective harmonics, we have

$$W_L[m] = \sum_{h=0}^{H-1} a_h^L \cos(2\pi hm/H + \theta_h^L)$$
$$W_R[m] = \sum_{h=0}^{H-1} a_h^R \cos(2\pi hm/H + \theta_h^R) \tag{11}$$

where a_h^L and θ_h^L are the amplitude and phase of the h^{th} harmonic in the left wavetable, and a_h^R and θ_h^R are defined analogously for the right wavetable.

Substituting (11) in (5) we obtain

$$W_{(n)}^F[m] = c(n) \sum_{h=0}^{H-1} a_h^L \cos(2\pi hm/H + \theta_h^L) + d(n) \sum_{h=0}^{H-1} a_h^R \cos(2\pi hm/H + \theta_h^R) \tag{12}$$

Assuming the corresponding harmonics in W_L and W_R are in phase (*i.e.* $\theta_h^L = \theta_h^R = \theta_h$) we have

---

[3]Depending on the class of instrument, the frequency register, and others factors like pitch and loudness, natural tones show different degrees of initial phase shift [24]. As is often done in additive synthesis and the phase vocoder [32], we ignore the initial phases, at least for most of Section 5. We do, however, consider initial phase differences in Section 5.7.

$$W^{E}_{(n)}[m] = \sum_{h=0}^{H-1} (c(n)a^{L}_{h} + d(n)a^{R}_{h}) \cos(2\pi hm/H + \theta_h) \tag{13}$$

Thus, the effective amplitude of the h^{th} harmonic in the effective wavetable $W^{E}_{(n)}$ is

$$a_h(n) = c(n)a^{L}_{h} + d(n)a^{R}_{h}. \tag{14}$$

Thus, given the constraints of spectral interpolation, the amplitude of a harmonic of the output at sample n is equal to the linear combination of the amplitude of the respective harmonic in the left and right wavetables, the combination coefficients being $c(n)$ and $d(n)$ as expected.

In this section we have described the process of generating an acoustic signal whose spectral composition can be dynamically modified through the interpolation of waveforms. In the next section we address the question of the analysis and reconstruction of an acoustic signal using such a model.

4. The Reconstruction Problem

Spectral interpolation can be used to synthesize a large class of instrumental sounds. We will discuss the types of sounds that are suitable for this type of synthesis and then address the problem of deriving control parameters by automatic analysis techniques.

4.1. Reproducing Harmonic Sounds

In section (3.3) we showed that a table lookup oscillator controlled with slowly-varying functions can generate only harmonic sounds. The same result applies to the waveform interpolation oscillator.[4] Thus, we will be able to reproduce only harmonic sounds.

What additional properties should a harmonic signal exhibit to make possible a quality resynthesis by waveform interpolation? Clearly, interpolation will best reproduce a signal with a slowly varying short-time spectrum. As a large class of instruments show gradual changes in their short-time spectra (especially in their sustain portions) we think that waveform interpolation is profitable in many situations.

This does not mean that it is not possible to reproduce signals with large, rapid spectral changes using waveform interpolation. Equation (7) may be flexible enough to adapt to such cases, as it allows the modification of the mixing coefficients of the wavetables, and switching of the waveforms themselves. A large control data bandwidth would be necessary to handle rapid spectral modulations.

4.2. Role of the Analysis

To reproduce a sound with the generation model described by the equation (7), we perform a sequence of interpolations between pairs of waveforms. The waveforms $\{W_i\}$ used in equation (7) are called *generator wavetables*. Each generator wavetable corresponds to a single period of the signal. Because of the interpolation function, the number of generator wavetables will be less than the total

---

[4] This is not exactly true. It is possible to generate sounds with inharmonic partials by interpolation between waveforms containing out-of-phase harmonics [10], and this effect may be used, for example, to generate vibrato [33]. However, as we discuss in section (3.6), it is not likely that we can systematically use this effect to reproduce spectral variations of a given sound, and by excluding this effect we accrue advantages in both analysis and synthesis (see section 3.6).

number of periods in the signal. The time it takes for the interpolation to go from one generator wavetable to the next ($(n_{i+1}-n_i)/srate$) is called the *interpolation interval*. During the synthesis, the periods located over an interpolation interval are computed with the interpolation formula (7). The function of the analysis preceding the synthesis is to select the generator wavetables, interpolation intervals, and weighting functions ($c(n)$ and $d(n)$) such that the resynthesized signal sounds like the original.

4.3. Non-Automatic Derivation of the Input Parameters in the Time Domain

One method of analysis consists in extracting the generator wavetables manually from periods of the original signal. In this method, the user studies a display of the signal over time, and selects the periods he believes to be the onset of significant spectral changes. These periods (each stretched to fill exactly M points) will be the generator wavetables, and the intervals between successive selected periods, the interpolation intervals. Time-linear interpolation may be used (in which $c(n)$ and $d(n)$ are determined from the interpolation intervals), or the user may specify (by, say, drawing) the mixing functions directly, subject to equation (9).

There are several inconveniences associated with manually selecting waveform interpolation parameters:

- The observation and interpretation of the time-domain evolution of the signal is laborious. This is due to the large amount of information displayed, and the difficulty of intuiting significant spectral changes from time-domain functions.
- Even in the simple case of time-linear interpolation, the process of extracting the generator wavetables manually from the original signal is inherently trial-and-error. Periods are selected, a signal is synthesized based on the selection, the result compared to the original, and the process is repeated until satisfactory results are obtained. In theory, this process has to be repeated for each new sound to be synthesized.
- If one desires to manually generate mixing functions as well, the problem becomes even more tedious, due to the interaction of mixing functions and generator wavetables on the synthesized signal.
- Further inconvenience occurs if the user wants to achieve as much data reduction as possible. In this case, when the resynthesized signal sounds good, the user is not sure that the number of generator wavetables can be decreased, and if the resynthesized signal is unsatisfactory, the user is not sure if he is using too few generator wavetables or just the wrong ones.
- When the waveforms are directly extracted from the original input signal, there may be phase differences among harmonics of the same order in successive generator waveforms. The resulting phase cancellation during resynthesis is hard to predict, making the selection process less intuitive. This problem can be somewhat alleviated by augmenting the manual selection process with some automatic phase analysis.

Non-automatic analysis is clearly difficult and labor-intensive. Thus, our effort has focused on building a general analysis model that computes the amplitudes and phases of the harmonics, then automatically selects (or computes) the generator wavetables, interpolation intervals, and weighting functions.

5. Analysis/Synthesis by Spectral Interpolation

In this section we present the analysis algorithm that precedes the spectral interpolation synthesis. The analysis algorithm takes as input the acoustic signal that we wish to regenerate, and outputs the control data for the synthesis.

In order to extract from the acoustic signal the relevant information for driving the synthesis, we follow several consecutive steps: digital recording of the sound, spectral analysis of the digitized sound, and data reduction. At the end of the analysis we arrive at a set of data describing (to some approximation) the original sound according to the spectral interpolation model. This set is then fed into the waveform interpolation synthesizer (or its software simulation) to verify the analysis.

5.1. Digital Recording

For the purpose of analysis we start to work with isolated tones played (as nearly as possible) at a constant pitch. These restrictions allow us to study separately the reproduction of the amplitude spectral variations by spectral interpolation. The analysis and synthesis described here will still work for tones whose pitch is not constant. However, since vibrato and connected pitches are facets of the tone we wish to control explicitly, we handle these at a higher level. This is briefly mentioned in Section 7 and will be described in a forthcoming paper.

The sound coming from the instrument is recorded using a microphone, and then sent to an analog/digital converter that transforms the analog signal into a stream of 16-bit samples. The sample rate and the anti-aliasing filter must be chosen according to the bandwidth of the signal. In addition we choose a sample rate that leads to an integer number of samples per period. For a given pitch, there are usually several possible sample rates in the system that can be suitable. We try to keep the sample rate close to 16 kHz, as we low-pass filter the signal at 6.4 kHz. If it is not practical to record at arbitrary sample rates, an interpolation and decimation procedure [16] or other resampling algorithm [38, 47] can change the sample rate efficiently. In order to fully experiment with our technique, we record several groups of notes for a given instrument. Each group contains notes played at constant pitch, and with different loudness and duration attributes. Thus we get different kinds of spectral evolutions to reproduce with the spectral interpolation model.

5.2. Spectral Analysis

There exist several methods of harmonic extraction based on the Short-Time Fourier Transform (phase vocoder, heterodyne filter, DFT). We use a pitch-synchronous DFT because it is simple and efficient. In addition, when the period is measured accurately, the DFT directly produces the amplitude and phase of each harmonic. Since we recorded the tone at a sample rate to insure an integral period, we need only to check that the period is correct. We do this using Moorer's optimum comb [30] and/or a simple peak detector.

Once the pitch modulations have been tracked and recorded, we can measure the evolution of the harmonics. For simplicity of presentation, we assume constant period P over the entire tone. The Discrete Fourier Transform of the i^{th} period of the discrete signal $x(n)$ is defined by:

$$X_k^{<i>} = \sum_{n=0}^{P-1} x(iP+n)\, e^{-j2\pi kn/P} \qquad 0 \le k \le P-1, \qquad 0 \le i < N_p \tag{15}$$

where P is the length of the period in number of samples, and N_p is the total number of periods in the tone. Equation (15) is equivalent to taking the Short-Time Fourier Transform of the signal $x(n)$ by using a rectangular window of duration P, and sampling it every P samples.

The complex DFT at period i, $X_h^{<i>}$, can be expressed in terms of its real and imaginary parts:

$$X_h^{<i>} = R_h^{<i>} + j I_h^{<i>}$$

The amplitudes of the DFT, $a_h^{<i>}$, are given by

$$a_h^{<i>} = |X_h^{<i>}| = \mathrm{sqrt}((R_h^{<i>})^2 + (I_h^{<i>})^2)$$

. The phase $\theta_h^{<i>}$ is given by

$$\theta_h^{<i>} = \mathrm{atan}\left(\frac{I_h^{<i>}}{R_h^{<i>}}\right)$$

As the signal $x(n)$ is real, the DFT amplitudes are symmetric: $|X_h^{<i>}| = |X_{P-h}^{<i>}|$. Thus, we have at most $\lfloor (P-1)/2 \rfloor$ harmonics (ignoring the DC component at $h=0$). In actuality, we calculate H harmonics, $H \le \lfloor (P-1)/2 \rfloor$, possibly ignoring some higher harmonics. We ignore the higher harmonics for a number of reasons: they often have insignificant amplitudes, the pitch-synchronous DFT computes them inaccurately, and we want to avoid aliasing when the tone is resynthesized at higher pitches. For simplicity, we evaluate the sum (equation 15) directly for each harmonic h, $1 \le h \le H$. This is often more efficient than using the FFT to compute the entire DFT, since H may be significantly less than $P/2$ (so there is no need to calculate all $P/2$ harmonics), and P is in general not a power of two (making FFT methods awkward).

The filter interpretation of the short-time Fourier transform [37] shows that if the DFT is computed on a sequence whose length equals the period (or a multiple of the period), then the amplitudes a_h represent the exact amplitudes of the harmonics. If the input is not perfectly harmonic, or if the period is not an integral number of samples, there will be some error due to the fact that in the passband of the filter for harmonic h (centered at $2\pi h/P$), more than one harmonic may be present, or the harmonic under analysis may not be in the center of the band. Nonetheless, we have obtained good results using the direct computation of equation (15). Also, we have been satisfied with the low time sampling rate of the short-time Fourier transform (one measure every P samples).

The DFTs result in a vector of amplitudes on each period of the tone. We call $S^{<i>}$ the spectrum measured at period i.

$$S^{<i>} = (a_1^{<i>}, a_2^{<i>}, \ldots, a_h^{<i>}, \ldots, a_H^{<i>}) \qquad (16)$$

The list of DFT spectra with their time indices is $\{(n_0, S^{<0>}), (n_1, S^{<1>}), \ldots, (n_{N_p-1}, S^{<N_p-1>})\}$. We call this list the "spectral envelope" of the tone. To reproduce the tone using the spectral interpolation model, we want to transform the list of DFT spectra into suitable data for the waveform interpolation synthesizer. That is the subject of the next section.

5.3. Spectral Paths and Spectral Ramps

For the remainder of section 5 we describe several algorithms that process the time-ordered list of DFT spectra $[(n_0,S^{<0>}),(n_1,S^{<1>}),\ldots,(n_{N_F-1},S^{<N_F-1>})]$. The purpose of each algorithm is to obtain the control data needed to drive the spectral-interpolation oscillator (equations (16), (11), (7)): a list of Q spectra (with associated times) and the scale functions $c(n)$ and $d(n)$. The number of spectra used for the synthesis Q should be much less than the number of spectra coming from the Fourier analysis N_F.

In order to explain how the data reduction works, we want to express the interpolation between two spectra $S^{<i>}$ and $S^{<j>}$ (successive in the synthesis) in terms of their individual harmonics.[5] Analogously to equation (14), we can express the instantaneous interpolated harmonics at sample n, $a_h^{<ij>}(n)$, as[6]

$$a_h^{<ij>}(n) = c(n)a_h^{<i>} + d(n)a_h^{<j>} \qquad n_i \le n < n_j \qquad 1 \le h \le H \tag{17}$$

The effective spectrum at sample n, $S^{<ij>}(n)$, is

$$S^{<ij>}(n) = c(n)S^{<i>} + d(n)S^{<j>} \qquad n_i \le n < n_j \tag{18}$$

We call the sequence of spectra $S^{<ij>}(n)$, $n_i \le n < n_j$, the *spectral path* from $S^{<i>}$ to $S^{<j>}$. A spectral path consists of the H loci which connect each harmonic of the initial spectrum $S^{<i>}$ to the harmonic of same order in the final spectrum $S^{<j>}$. A spectral path is defined by a set of two amplitude spectra (H amplitude values for each spectrum), and a mixing function defined by the two coefficients $c(n)$ and $d(n)$, for $n_i \le n < n_j$.

The interpretation of a spectral path is straightforward when the interpolation is linear in time. In this case, the harmonic amplitudes (equation (10)) are given by:

$$S^{<ij>}(n) = (1-sp(n))S^{<i>} + sp(n)S^{<j>} \qquad n_i \le n < n_j$$

$$a_h^{<ij>}(n) = (1-sp(n))a_h^{<i>} + sp(n)a_h^{<j>} \qquad 1 \le h \le H \qquad n_i \le n < n_j$$

$$sp(n) = \frac{n-n_i}{n_j-n_i} \tag{19}$$

Equation (19) is equivalent to:

$$a_h^{<ij>}(n) = a_h^{<i>} + \frac{n-n_i}{n_j-n_i}(a_h^{<j>} - a_h^{<i>}) \qquad 1 \le h \le H \qquad n_i \le n < n_j \tag{20}$$

Here, the effect of interpolating between two spectra is that the amplitude of each harmonic ramps linearly from its value in the first spectrum to its value in the second. For this reason $S^{<ij>}$ (consisting of H amplitude ramps) is called a *spectral ramp*. A *spectral ramp* is defined by the set of H initial and H final values of the harmonic amplitudes, together with the duration of the interpolation (n_j-n_i).

Figure 5-1 shows three successive spectral ramps. Representing an amplitude spectrum evolution

---

[5] This explanation assumes that the spectra used in the synthesis have been selected from the DFT spectra computed in the analysis. While this is often the case, one of our algorithms (see section 5.4.2) computes the spectra used in the synthesis.

[6] For the remainder of section 5 we assume without loss of generality that the waveform for $S^{<i>}$ is in the left wavetable and the waveform for $S^{<j>}$ is in the right. If this is not the case, as happens every other pair of spectra, the roles of $c(n)$ and $d(n)$ must be interchanged. The equations as written generate sawtooth-like functions for $c(n)$ and $d(n)$ which after the interchange become the triangle-like functions as constrained by equation (9).

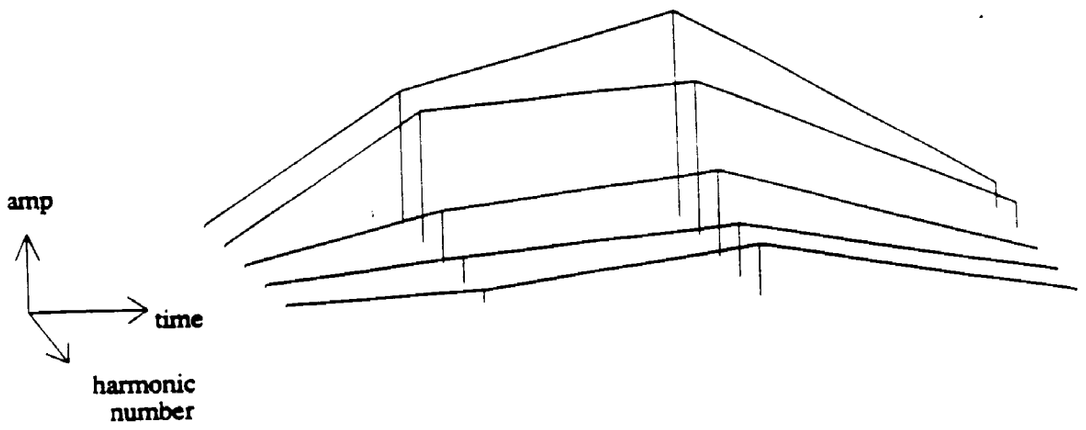

Figure 5-1: Three Spectral Ramps

with spectral ramps is similar to using a piecewise linear approximation for the individual harmonic amplitudes. However, in contrast to the usual representations used for additive synthesis [21], the breakpoints that define the piecewise linear function for each harmonic are simultaneous.

In order to ensure that the reproduced spectra are close to the original spectra, the data reduction algorithm is based on an error-minimization process. The various methods we use to determine the spectral paths in order to minimize our error criteria are described in the following sections.

5.4. Data-Reduction using the Time Linear Spectral Interpolation Representation

We first consider using spectral ramps for resynthesis. The fitting of the spectral envelope with a small set of spectral ramps is based on the following process: starting with the spectrum of the first period of the tone, we compute the spectral ramp to the spectrum of each successive period in turn. For each successive period we calculate an error measure based on the deviation between the harmonic amplitudes of the original spectra and those of the computed spectral ramp. When the error exceeds a given threshold, the spectral ramp ending at the previous period is stored. The process is repeated using the end of this spectral ramp as the initial spectrum of the next spectral ramp. This loop is executed until the entire spectral envelope has been approximated.

We have tried two different ways for optimizing the spectral ramps. The first method is called *spectral ramp interpolation using original spectra*. It selects some of the DFT spectra in the spectral envelope of the original tone as endpoints of the spectral ramps. The second method is called *spectral ramp interpolation using computed spectra*. It uses a linear regression algorithm to compute the spectral ramps. We discuss each of these in turn.

5.4.1. Spectral Ramp Interpolation Using Original Spectra

Consider equation (19) which defines an interpolated spectrum $S^{<ijP>}(n)$ on each sample n within the ramp $S^{<ijP>}$. For the purpose of measuring the error on the spectral ramp $S^{<ijP>}$, we need to compare the successive computed spectra $S^{<ijP>}(n)$, $n_i \le n < n_j$ to the corresponding sequence of DFT spectra $S^{<l>}$ with $i \le l < j$ (since $n_i = iP$ and $n_j = jP$). Thus we need to sample the computed spectra at the same rate as the

DFT spectra, *i.e.* with a time interval equal to the period P. We consider that the interpolated spectrum on period l is equal to the spectrum computed at sample $n_l = lP$, i.e $S^{<i,P>}<l> = S^{<i,P>}(n_l)$. Given this notation equation (20) can be rewritten:

$$a_h^{<i,P>}<l> = a_h^{<i>} + \frac{n_l - n_i}{n_j - n_i}(a_h^{<P>} - a_h^{<i>}) \qquad i \le l < j \tag{21}$$

Comparing the amplitudes of the harmonics of spectrum $S^{<l>}$ with values given by equation (21) produces an error $E_{<l>}^{<i,P>}$. $E_{<l>}^{<i,P>}$ is defined as the mean square error on the H harmonic amplitudes:[7]

$$E_{<l>}^{<i,P>} = \sum_{h=1}^{H} (a_h^{<i,P>}<l> - a_h^{<l>})^2 \tag{22}$$

Using the following notation:

$$\Delta a_h^{ij} = a_h^{<P>} - a_h^{<i>} \qquad \Delta a_h^{il} = a_h^{<l>} - a_h^{<i>} \qquad \Delta n^{ij} = n_j - n_i \qquad \Delta n^{il} = n_l - n_i \tag{23}$$

equation (21) becomes:

$$a_h^{<i,P>}<l> = a_h^{<i>} + \frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{ij} \tag{24}$$

From equations (24) and (22) we deduce:

$$E_{<l>}^{<i,P>} = \sum_{h=1}^{H} (\frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{ij} + a_h^{<i>} - a_h^{<l>})^2$$

$$= \sum_{h=1}^{H} (\frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{ij} - \Delta a_h^{il})^2 \tag{25}$$

The global error $E^{<i,P>}$ within the spectral ramp $S^{<i,P>}$ is defined as the sum of the errors on the individual spectra:[8]

$$E^{<i,P>} = \sum_{l=i+1}^{j-1} E_{<l>}^{<i,P>} \tag{26}$$

If the error $E^{<i,P>}$ is less than the tolerated threshold E_{max} we extend the spectral ramp to the next period <j+1> and compute the new error $E^{<i,j+1>}$ using equation (26), $E_{<l>}^{<i,j+1>}$ being computed with equation (25). Otherwise we store the data defining the previous the spectral ramp $S^{<i,j-1>}$, and we compute the next ramp starting at spectrum $(n_{j-1}, S^{<j-1>})$.

5.4.2. Spectral Ramp Interpolation Using Computed Spectra

The *Linear Regression* algorithm is a way of fitting piecewise linear functions to a set of points [51]. We use a variant of linear regression called *anchored regression* [40]. Given a fixed point called the *anchor* and a set of data points, the anchored regression algorithm finds the slope of the line passing through the anchor which minimizes the sum of squared distances from the data points to the line. We use this algorithm for the computation of the H segments which define the spectral ramp. Instead of processing each harmonic separately (which would probably result in a set of segments of different

---

[7] The averaging factor $1/H$ is omitted here as it is assumed constant for the tone under study.

[8] We may start the sum from $i+1$ since as we use $S^{<i>}$ as the initial spectrum in the spectral ramp, we have $E_{<i>}^{<i,P>} = 0$.

lengths) we perform H linear regressions on a fixed time interval, and we compute a global error on the resulting spectral ramp.

The anchors (or the endpoints of the spectral ramps) are notated $S'^{<i>}$. The prime is used to differentiate the anchors from the DFT spectra. The anchored regression algorithm works as follows: We start with an anchor $(n_i, S'^{<i>})$. For the first anchor we take $(n_0, S^{<0>})$[9]. For each successive consecutive DFT spectrum $(n_j, S^{<j>})$, $j > i$, we compute H lines. The h^{th} line goes through its anchor $(n_i, a'_h{<i>})$ and comes closest (in the least squares sense) to the set of points $(n_{i+1}, a_h{<i+1>}), \ldots, (n_j, a_h{<j>})$. The error on the spectral ramp is computed as the sum of the errors of the H individual lines. If the error is below a threshold E_{max} the next DFT spectrum $(n_{j+1}, S^{<j+1>})$ is examined. Otherwise the spectral ramp $S'^{<i,j-1>}$ is used. In this case the endpoint of the spectral ramp is (n_{j-1}, S'^{j-1}). The H coordinates of $S'^{<j-1>}$, $a'_h{<j-1>}$, for $1 \le h \le H$, are computed using the slopes of the H best lines. The endpoint $(n_{j-1}, S'^{<j-1>})$ is then used as the new anchor and the process is repeated to get the next spectral ramp.

Using a simple least squares fit, it is straightforward to compute the best spectral ramp. Since each of the H lines forming the spectral ramp $S'^{<ij>}$ must go through the coordinates $(n_i, a'_h{<i>})$ each line is defined by an equation of the form

$$a'^{<ij>}_h{<l>} = m^{<ij>}_h (n_l - n_i) + a'_h{<i>} \qquad 1 \le h \le H \qquad i \le l \le j \tag{27}$$

where $m^{<ij>}_h$ is the slope of the h^{th} line of the ramp.

Defining $E^{<ij>}_h$ to be the sum of the squared errors of each harmonic within the ramp $S'^{<ij>}$, we have

$$E^{<ij>}_h = \sum_{l=i+1}^{j} (a_h{<l>} - (m^{<ij>}_h (n_l - n_i) + a_h{<i>}))^2 \tag{28}$$

The total error on the spectral ramp $E^{<ij>}$ is the sum of the errors on each harmonic:

$$E^{<ij>} = \sum_{h=1}^{H} E^{<ij>}_h \tag{29}$$

By using the notation previously defined (equation (23)) we can write:

$$E^{<ij>}_h = \sum_{l=i+1}^{j} (\Delta a^{il}_h - m^{<ij>}_h \Delta n^{il})^2 \tag{30}$$

$$E^{<ij>} = \sum_{h=1}^{H} \left(\sum_{l=i+1}^{j} (\Delta a^{il}_h - m^{<ij>}_h \Delta n^{il})^2 \right) \tag{31}$$

Differentiating the error with respect to $m^{<ij>}_h$, we obtain

$$\frac{\partial E^{<ij>}}{\partial m^{<ij>}_h} = -2 \sum_{l=i+1}^{j} (\Delta a^{il}_h - m^{<ij>}_h \Delta n^{il}) \Delta n^{il} \tag{32}$$

Setting each derivative $\frac{\partial E^{<ij>}}{\partial m^{<ij>}_h} = 0$ gives the slope of each line:

---

[9] Here the prime is omitted because the anchor is not computed.

$$m_h^{<ij>} = \frac{\sum_{l=i+1}^{j} \Delta n^{il} \Delta a_h^{il}}{\sum_{l=i+1}^{j} (\Delta n^{il})^2} \tag{33}$$

As we mentioned before, if the total error $E^{<ij>}$ is less than the threshold, we add the next spectrum $S^{<j+1>}$ to our set, and calculate the error $E^{<i,j+1>}$. Interestingly, we do not have to reevaluate from scratch the sums in equations (31) and (33) when we add the new spectra. Rewriting (30) gives

$$E_h^{<ij>} = \sum_{l=i+1}^{j} (\Delta a_h^{il})^2 - 2 m_h^{<ij>} \sum_{l=i+1}^{j} \Delta n^{il} \Delta a_h^{il} + (m_h^{<ij>})^2 \sum_{l=i+1}^{j} (\Delta n_{il})^2 \tag{34}$$

Now, each of the sums in (33) and (34) may be computed incrementally:

$$\begin{aligned}
&\sigma_{nn}^{<ii>} = \sigma_{aa}^{<ii>}(h) = \sigma_{na}^{<ii>}(h) = 0 \\
&\sigma_{nn}^{<ij>} = \sigma_{nn}^{<i,j-1>} + (\Delta n^{j-1,j})^2 \\
&\sigma_{aa}^{<ij>}(h) = \sigma_{aa}^{<i,j-1>}(h) + (\Delta a_h^{j-1,j})^2 \\
&\sigma_{na}^{<ij>}(h) = \sigma_{na}^{<i,j-1>}(h) + \Delta a_h^{j-1,j} \Delta n^{j-1,j} \qquad \text{for } i < l \le j \quad \text{and} \quad 1 \le h \le H
\end{aligned} \tag{35}$$

Now, $m_h^{<ij>}$ (equation (33)) and $E_h^{<ij>}$ (equation (34)) can be computed as

$$m_h^{<ij>} = \frac{\sigma_{na}^{<ij>}(h)}{\sigma_{nn}^{<ij>}(h)}$$

$$E_h^{<ij>} = \sigma_{aa}^{<ij>}(h) - 2 m_h^{<ij>} \sigma_{na}^{<ij>}(h) + m_h^{<ij>2} \sigma_{nn}^{<ij>} \tag{36}$$

The incremental computation allows us to do a small amount of work (proportional to H) to compute the new slopes and error when adding a spectrum to the regression. In other words, we can compute every error $E^{<ii>}, E^{<i,i+1>}, \ldots, E^{<ij>}$ with the same effort as it takes to just compute $E^{<ij>}$.

5.4.3. First Results

After computing the spectral ramps (using either method), we resynthesize the tone by evaluating equations (11) and (10) in software. The successive spectra defining the spectral ramps are sent to the *waveform generator*. The waveform generator computes one cycle of a waveform from a given amplitude spectrum. One period of the waveform is obtained by adding H sine waves, each scaled by the corresponding amplitude. The phases of the sine waves are alternately set to 0 or π, so that the waveform and its first derivative are close to zero at the beginning and at the end of the table. This is the technique is used in the Bradford Musical Instrument Simulator [13]. After two waveforms defining a spectral ramp have been loaded in the wavetables, the wavetables are read, scaled by two opposite linear ramps (10) and added to form the output signal. The process is repeated until every spectral ramp has been synthesized.

The degree to which the synthesized signal approximates the input signal depends on the number of spectral ramps output by the data reduction algorithm. This number can be modified by varying the threshold E_{max}. For each tone, we run the algorithm several times using different thresholds. We then choose the synthesized signal with the smallest number of spectra (largest threshold) that is perceptually indistinguishable from the original tone. The chosen spectral ramps are an accurate (and usually succinct) representation of the tone.

We have obtained very good results on a number of instruments belonging to the woodwind and brass families (bassoon, clarinet, saxophone, trumpet, trombone). The average reduction rate on the number of spectra Q/N_p is on the order of 10%. The two algorithms, spectral ramp interpolation using original spectra and spectral ramp interpolation using computed spectra, were found to be almost identical in terms of the data reduction they achieve and in their computational cost. For an equivalent data reduction rate, the two algorithms lead to a similar perceptual output. Using computed spectra usually achieves a slightly greater data reduction, since by computing spectra a given amount of error can be spread over a longer spectral ramp than is possible using original spectra.

These promising results were obtained on sounds whose spectral evolution was rather regular. Figure 5-2 shows a sampled trombone tone[10]. Beneath the tone are the spectra which resulted from the DFT analysis of the tone. Under those are the spectra selected by the data reduction algorithm. The synthesized tone is shown under the selected spectra.

When applied to sounds whose amplitudes vary rapidly (as in a rapid tremolo), the data reduction was less effective, although always below 50%. In this kind of situation, a large number of spectral ramps were needed to track the changes in amplitude (and their corresponding spectral changes). As our algorithm proceeds sequentially without any global view of the signal, it does not take advantage of the oscillation between two close spectra, characteristic of the tremolo. To improve the data reduction (in these cases and in general), we have tried another method, based on *nonlinear interpolation*. By relaxing the constraint that the scaling factors $c(n)$ and $d(n)$ (equation (18)) be two opposite linear ramps summing to unity, we allow arbitrary combinations of the two waveforms. Thus instead of resorting to the computation of new waveforms, we look for more complex time-varying mixing functions, with the hope that these will result in greater data reduction. The nonlinear interpolation analysis is described in the next section.

5.5. Data-Reduction using the Nonlinear Spectral Interpolation Representation

In this section we consider that the paths connecting the harmonics (equation (17)) $c(n)$ and $d(n)$ are arbitrary functions of time. To determine the optimal set of spectral paths we use a similar procedure as presented in the previous section. But, instead of optimizing the spectra defining the ends of the spectral path, we select two spectra in the DFT list $S^{<i>}$ and $S^{<j>}$ separated by an arbitrarily large time interval, and we minimize the square error within the spectral path by choosing the best coefficients $c(n)$ and $d(n)$. If the error is larger than the threshold E_{max}, we try the same operation with a smaller interpolation duration, i.e we try the spectral path that links together the spectra $S^{<i>}$ and $S^{<j-1>}$, and so on.

Within the spectral path $S^{<ij>}$, the harmonics of the interpolated spectrum $S^{<ij>}_{<l>}$ at period l are computed

[10] Actually, part of the sustain portion of the tone was removed so we could fit the signal on the page.

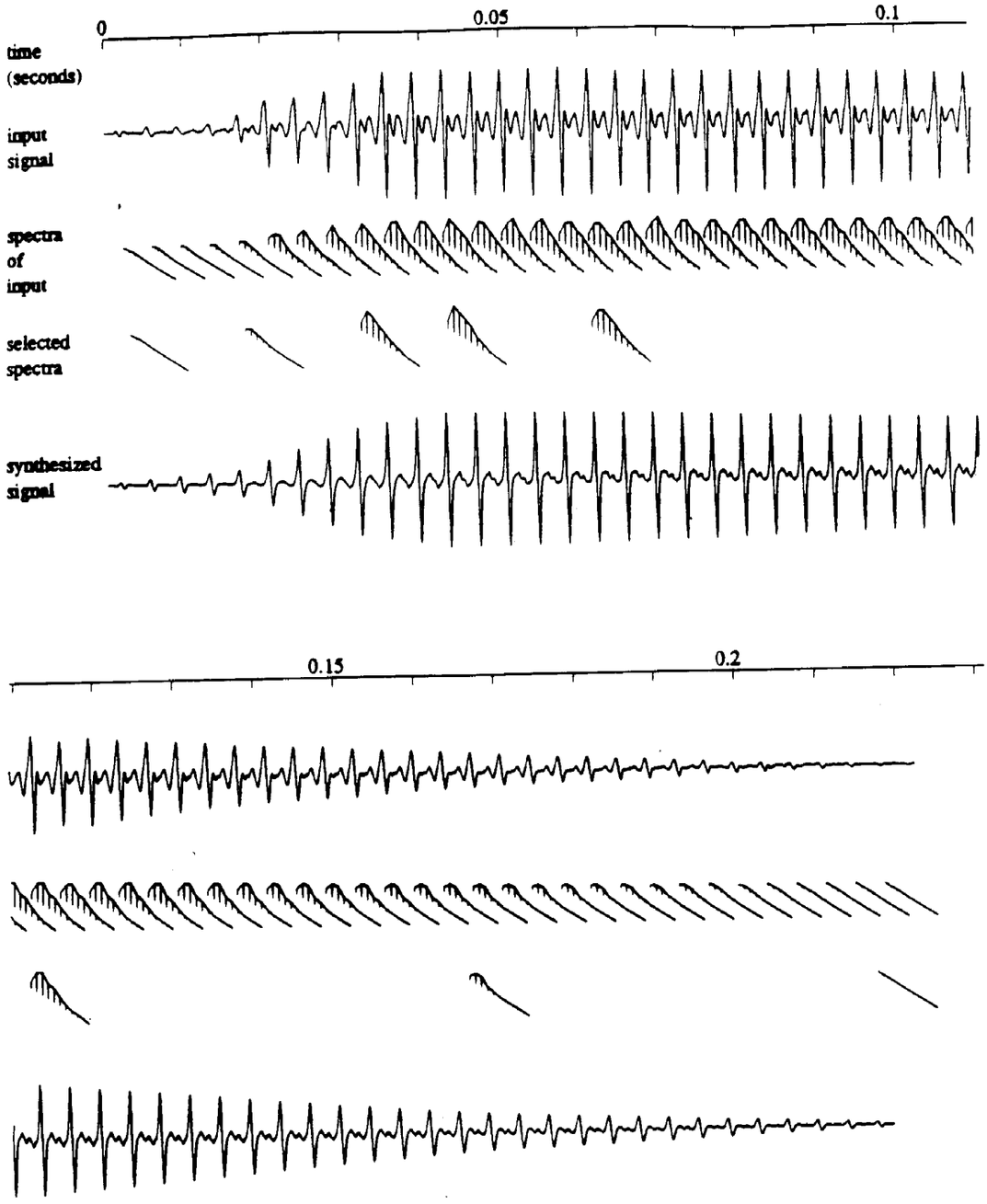

Figure 5-2: Synthesis by Time Linear Spectral Interpolation

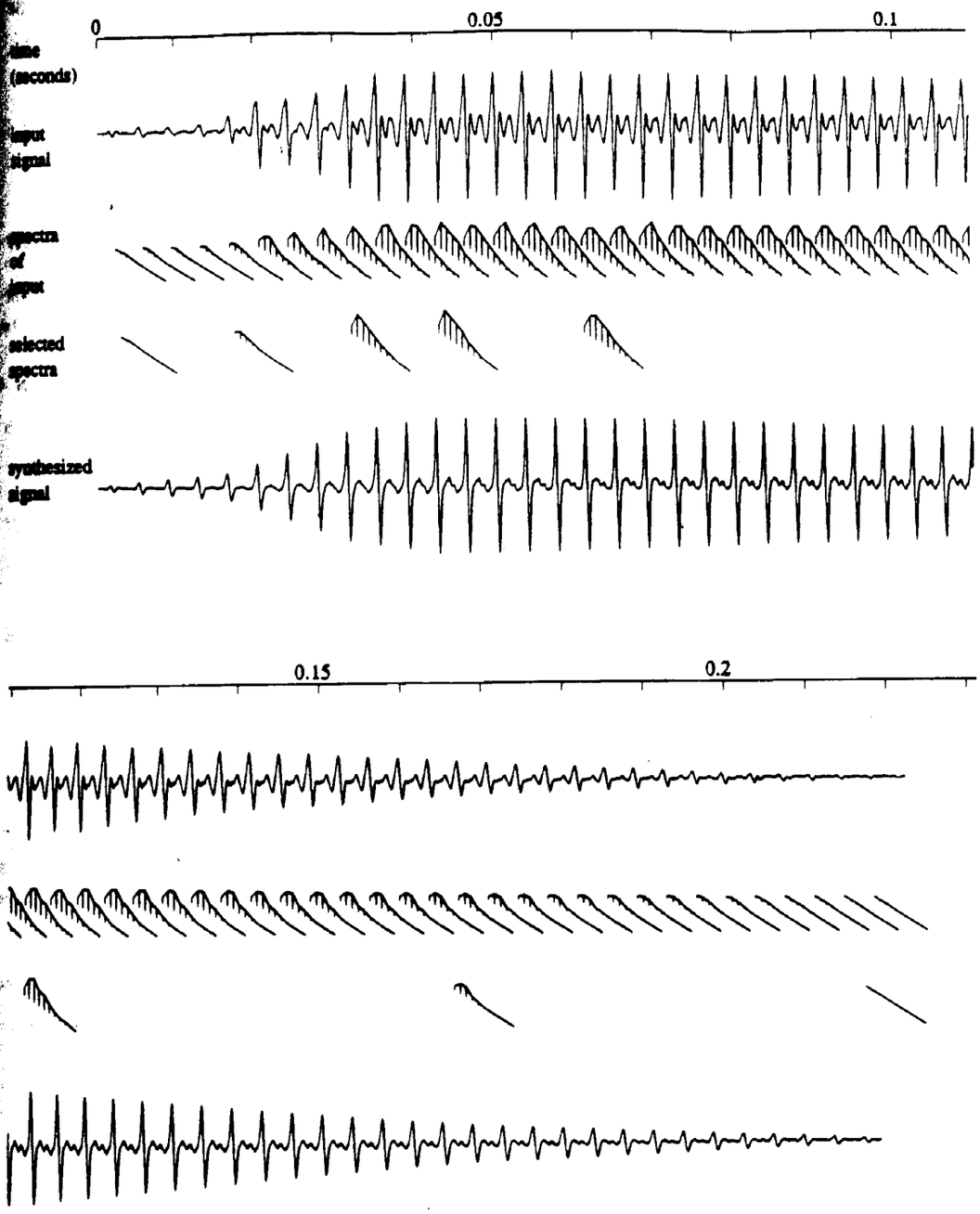

Figure 5-2: Synthesis by Time Linear Spectral Interpolation