Analysis and Synthesis of Tones by Spectral Interpolation*

MARIE-HELENE SERRA, DEAN RUBINE, AND ROGER DANNENBERG

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

A technique is presented for the analysis and digital resynthesis of instrumental sounds. The technique is based on a model that uses interpolation of amplitude spectra to reproduce short-time spectral variations. The main focus of this work is the analysis algorithm. Starting from a digital recording the authors were able to compute automatically the parameters of this 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. Two analysis/synthesis methods are studied based on spectral interpolation. The first uses only spectral interpolation. This method made it possible 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. The authors plan to extend the study to many more instruments.

0 INTRODUCTION

All approaches to the synthesis of musical tones represent a compromise among *generality*, the ability to produce any sound; *efficiency*, the computational cost of synthesis; and *control*, the ability to simply, flexibly, and intuitively control musical parameters such as timbre, pitch, and loudness. For example, additive synthesis exhibits a high degree of generality in that any sound can be created, but additive synthesis [1] is not efficient, requiring a large amount of computation and control information, and control is complex. FM synthesis [2] is more efficient but not as general as summation synthesis. The amount of control information is reduced, but the control parameters are not always intuitive, and automatic analysis of instrumental sounds to obtain FM parameters is largely an open problem.

Fixed waveform synthesis, waveshaping, subtractive synthesis, sampling, and physical modeling are other examples, each providing a different combination of generality, efficiency, and control, but no technique scores highly on all criteria. We have investigated a new technique that trades some generality for efficiency and control. The technique is inspired by both additive and fixed waveform synthesis, and control parameters can be derived automatically by analysis of natural sounds.

Sec. 1 presents a new technique, called spectral interpolation synthesis, and Sec. 2 describes the analysis/synthesis problem we studied. Sec. 3 presents and evaluates several solutions we explored. The remaining sections describe related work, directions for further study, and our conclusions.

1 SYNTHESIS BY SPECTRAL INTERPOLATION

Perhaps the least expensive way to generate tones is through the use of fixed-waveform (or table-lookup) synthesis, which uses one digital oscillator per voice [3]. Unfortunately, fixed-waveform synthesis does not offer the possibility of time-varying spectral content, which is important in the production of interesting musical tones.

One solution is to sum the outputs of a number of

* Manuscript received 1988 September 23. This work was supported by Apple Computer, Inc., and the Yamaha Corporation. Further support was provided by an IBM Fellowship Supplement; a Ben Franklin Partnership Fund Matching Award, Agreement 402995-45176-308-001; and the Defense Advanced Research Projects Agency (DOD), ARPA order 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, OH 45433-6543.

fixed-waveform oscillators. The amplitude controls can then be used to control various parts of the spectrum independently. In the limit, each oscillator produces a sinusoidal partial, and we have summation synthesis. This approach is expensive because many oscillators are needed for each voice or musical tone.

Another approach is to change the waveform in a single oscillator. If the waveform currently addressed is changed instantaneously 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 [4]. Unfortunately, avoiding perceptual discontinuities (clicks) at the switching points requires a large amount of input data [5], [6]. 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 wave tables.

Fig. 1 illustrates the waveform interpolation oscillator. Waveform interpolation is similar in implementation to table lookup but uses two phase-locked wave tables, W_L and W_R, each loaded with a different waveform.

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

$$\text{phase}(n) = \text{phase}(n-1) + \text{phaseInc}(n) \quad [\text{mod } M] \tag{1}$$

$$y(n) = c(n)W_L[\text{phase}(n)] + d(n)W_R[\text{phase}(n)] \tag{2}$$

where

n	=	number (index) of sample being computed
M	=	(constant) number of samples in wave table W
phaseInc(n)	=	phase increment at sample n
$W[m]$	=	mth sample of fixed wave table W
$c(n)$	=	amplitude scale factor of left wave table at sample n
$d(n)$	=	amplitude scale factor of right wave table at sample n
phase(n)	=	phase accumulator at sample n
$y(n)$	=	output signal at sample n.

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 alternative view is to consider a waveform interpolation oscillator to be a table-lookup oscillator whose "effective" table is computed dynamically as the linear combination of two fixed wave tables,

$$W^E_{(n)}[m] = c(n)W_L[m] + d(n)W_R[m] \; . \tag{3}$$

1.1 Arbitrary Spectral Evolution via Waveform Interpolation

We would like to use waveform interpolation [Eq. (2)] to generate a spectral change over time. This is accomplished by switching the wave tables themselves over the course of a sound. For instance, by repeating the interpolation procedure [Eq. (2)] 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, only one of the two waveforms is changed at any one time, and the change occurs when the scaling factor associated with the wave table being changed is zero.

Fig. 2 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), . . ., (W_{Q-2}, W_{Q-1}).

In the figure we have shown $c(n)$ and $d(n)$ as piecewise linear functions, each of which alternatively 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 wave tables change.

1.2 Spectral Interpolation

Problems may occur when interpolating between two waveforms whose phase distributions are different. Phase cancellation causes the amplitude of each harmonic in the interpolated signal to be less than expected [3]. 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

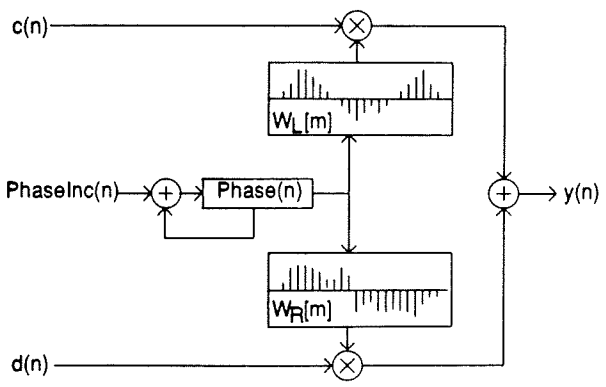

Fig. 1. Waveform interpolation oscillator.

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 used extensively 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 [7]. In addition to ignoring phase shifts of a given harmonic in time, we ignore the initial phase differences between different harmonics.[1] We use the term *synthesis by spectral interpolation* to mean that we have constrained the corresponding harmonics of each generator wave table to be in phase.

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

$$W_L[m] = \sum_{h=0}^{H-1} a_h^L \cos\left(\frac{2\pi hm}{H} + \theta_h^L\right)$$
$$W_R[m] = \sum_{h=0}^{H-1} a_h^R \cos\left(\frac{2\pi hm}{H} + \theta_h^R\right) \quad (4)$$

where a_h^L and θ_h^L are the amplitude and the phase of the hth harmonic in the left wave table, and a_h^R and θ_h^R are defined analogously for the right wave table.

Substituting Eq. (4) in Eq. (3) we obtain

$$W_{(n)}^E[m] = c(n) \sum_{h=0}^{H-1} a_h^L \cos\left(\frac{2\pi hm}{H} + \theta_h^L\right) + d(n) \sum_{h=0}^{H-1} a_h^R \cos\left(\frac{2\pi hm}{H} + \theta_h^R\right). \quad (5)$$

Assuming that the corresponding harmonics in W_L and W_R are in phase (that is, $\theta_h^L = \theta_h^R = \theta_h$), we have

$$W_{(n)}^E[m] = \sum_{h=0}^{H-1} (c(n)a_h^L + d(n)a_h^R) \cos\left(\frac{2\pi hm}{H} + \theta_h\right). \quad (6)$$

Thus the effective amplitude of the hth harmonic in the effective wave table $W_{(n)}^E$ is

$$a_h(n) = c(n)a_h^L + d(n)a_h^R . \quad (7)$$

Given the constraints of spectral interpolation, the amplitude of a harmonic of the output at sample n is thus equal to the linear combination of the amplitude of the respective harmonic in the left and right wave tables, 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. Next we address the question of the analysis and reconstruction of an acoustic signal using such a model.

2 THE RECONSTRUCTION PROBLEM

Spectral interpolation, achieved by interpolating between successive waveforms, can be used to synthesize a large class of instrumental sounds. If we use slowly

[1] Depending on the class of instrument, the frequency register, and other factors such as pitch and loudness, natural tones show different degrees of initial phase shift [8]. As is often done in additive synthesis and the phase vocoder [9], we ignore the initial phases, at least for most of Sec. 3. We do, however, consider initial phase differences in Sec. 3.7.

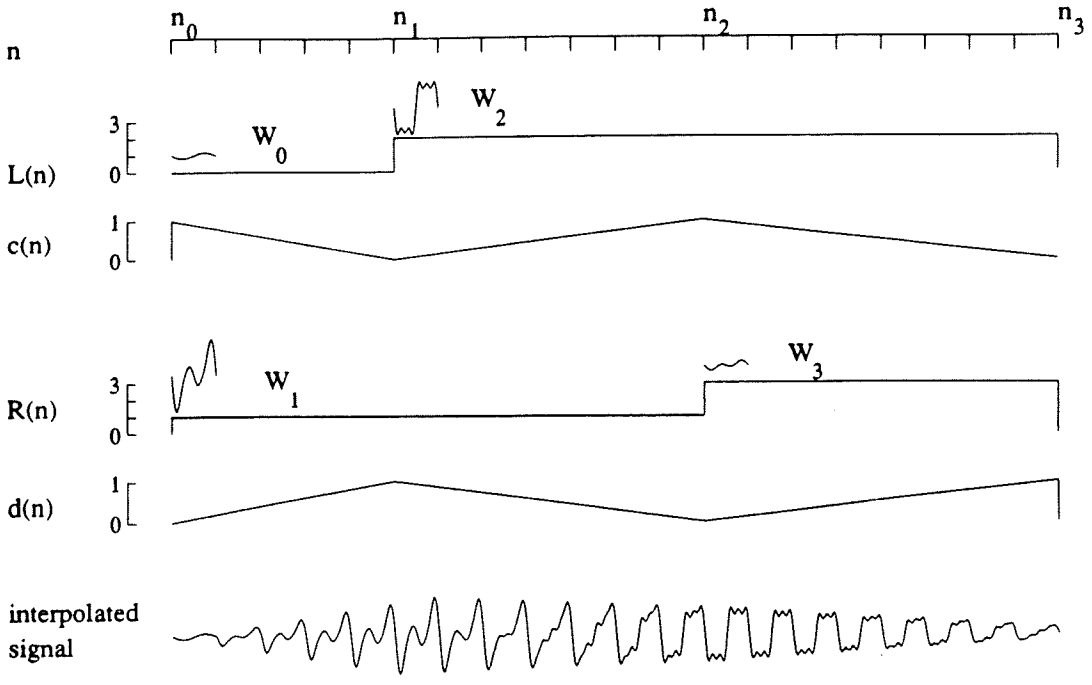

Fig. 2. Interpolation of successive waveforms.

changing control functions to control amplitude and frequency, then the waveform interpolation oscillator will produce only sounds with harmonically related partials.[2] This is not appropriate for many percussion and noise sounds, but excellent results have been obtained for wind instrument sounds. To produce realistic instrument sounds, it is most helpful to analyze sounds in order to obtain control parameters for the spectral interpolation model described by the following equation:

$$y(n) = c(n)W_{L(n)}[\text{phase}(n)] + d(n)W_{R(n)}[\text{phase}(n)] \quad (8)$$

where $L(n)$ and $R(n)$ determine the left and right waveforms at sample n.

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. Eq. (8) 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.

2.1 Role of the Analysis

To reproduce a sound with the generation model described by Eq. (8), we perform a sequence of interpolations between pairs of waveforms stored as the set of wave tables $\{W_i\}$. Each wave table corresponds to a single period of the signal. Because of the interpolation function, the number of wave tables will be less than the total number of periods in the signal. The time it takes for the interpolation to go from one wave table to the next is called the *interpolation interval*. The function of the analysis preceding the synthesis is to select the wave tables, interpolation intervals, and weighting functions [$c(n)$ and $d(n)$] such that the resynthesized signal sounds like the original.

2.2 Nonautomatic Derivation of Input Parameters in the Time Domain

One method of analysis consists in extracting the wave tables 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 or she believes to be the onset of significant spectral changes. These periods (each stretched to fill exactly M points) will be the wave tables, 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.

Nonautomatic 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 wave tables, interpolation intervals, and weighting functions.

3 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.

3.1 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 Sec. 5 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-to-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 [11] or other resampling algorithm [12], [13] can change the sample rate efficiently. In order to fully

[2] This is not exactly true. It is possible to generate sounds with inharmonic partials by interpolation between waveforms containing out-of-phase harmonics [3], and this effect may be used, for example, to generate vibrato [10]. However, as we discuss in Sec. 1.2, 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 Sec. 1.2).

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.

3.2 Spectral Analysis

There exist several methods of harmonic extraction based on the short-time Fourier transform [phase vocoder, heterodyne filters, discrete Fourier transform (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 ensure an integral period, we need only to check that the period is correct. We do this using Moorer's optimum comb [14] 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 DFT of the ith period of the discrete signal $x(n)$ is defined by

$$X_h^{\langle i \rangle} = \sum_{n=0}^{P-1} x(iP + n)e^{-j2\pi hn/P},$$

$$0 \leq h \leq P - 1, \quad 0 \leq i < N_P \quad (9)$$

where P is the length of the period in number of samples and N_P is the total number of periods in the tone. Eq. (9) 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.

As the signal $x(n)$ is real, the DFT amplitudes are symmetric: $|X_h^{\langle i \rangle}| = |X_{P-h}^{\langle i \rangle}|$. Thus we have at most $[(P - 1)/2]$ harmonics (ignoring the dc component at $h = 0$). In actuality, we calculate H harmonics, $H \leq [(P - 1)/2]$, 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 evalute the sum [Eq. (9)] directly for each harmonic h, $1 \leq h \leq H$. This is often more efficient than using the fast Fourier transform (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 2 (making FFT methods awkward).

The filter interpretation of the short-time Fourier transform [15] 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 Eq. (9). 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 $S^{\langle i \rangle}$ the spectrum measured at period i,

$$S^{\langle i \rangle} = (a_1\langle i \rangle, a_2\langle i \rangle, \ldots, a_h\langle i \rangle, \ldots, a_H\langle i \rangle) \quad (10)$$

where $a_h\langle i \rangle = |X_h^{\langle i \rangle}|$, that is, the magnitude of the hth harmonic at period i.

The list of DFT spectra with their time indices is $\{(n_0, S^{\langle 0 \rangle}), (n_1, S^{\langle 1 \rangle}), \ldots, (n_{N_P-1}, S^{\langle N_P-1 \rangle})\}$. 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.

3.3 Spectral Paths and Spectral Ramps

For the remainder of Sec. 3 we describe several algorithms that process the time-ordered list of DFT spectra $\{(n_0, S^{\langle 0 \rangle}), (n_1, S^{\langle 1 \rangle}), \ldots, (n_{N_P-1}, S^{\langle N_P-1 \rangle})\}$. The purpose of each algorithm is to obtain the control data needed to drive the spectral-interpolation oscillator [Eqs. (10), (4), (8)]: 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_P.

In order to explain how the data reduction works, we want to express the interpolation between two spectra $S^{\langle i \rangle}$ and $S^{\langle j \rangle}$ (successive in the synthesis) in terms of their individual harmonics.[3] Analogously to Eq. (7), we can express the instantaneous interpolated harmonics at sample n, $a_h^{\langle ij \rangle}(n)$, as[4]

$$a_h^{\langle ij \rangle}(n) = c(n)a_h^{\langle i \rangle} + d(n)a_h^{\langle j \rangle},$$

$$n_i \leq n < n_j, \quad 1 \leq h \leq H . \quad (11)$$

The effective spectrum at sample n, $S^{\langle ij \rangle}(n)$, is

$$S^{\langle ij \rangle}(n) = c(n)S^{\langle i \rangle} + d(n)S^{\langle j \rangle},$$

$$n_i \leq n < n_j . \quad (12)$$

[3] 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 Sec. 3.4.2) computes the spectra used in the synthesis.

[4] For the remainder of Sec. 3 we assume without loss of generality that the waveform for $S^{\langle i \rangle}$ is in the left wave table and the waveform for $S^{\langle j \rangle}$ 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 sawtoothlike functions for $c(n)$ and $d(n)$, which after the interchange become the trianglelike functions as illustrated in Fig. 2.

We call the sequence of spectra $S^{(ij)}(n)$, $n_i \leq n < n_j$, the *spectral path* from $S^{(i)}$ to $S^{(j)}$. A spectral path consists of the H loci that connect each harmonic of the initial spectrum $S^{(i)}$ to the harmonic of the 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 $n_i \leq n < n_j$.

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

$$S^{(ij)}(n) = [1 - \mathrm{sp}(n)]S^{(i)} + \mathrm{sp}(n)S^{(j)}, \quad n_i \leq n < n_j$$

$$a_h^{(ij)}(n) = [1 - \mathrm{sp}(n)]a_h^{(i)} + \mathrm{sp}(n)a_h^{(j)}, \quad 1 \leq h \leq H, \quad n_i \leq n < n_j$$

$$\mathrm{sp}(n) = \frac{n - n_i}{n_j - n_i}. \tag{13}$$

Eq. (13) is equivalent to

$$a_h^{(ij)}(n) = a_h^{(i)} + \frac{n - n_i}{n_j - n_i}(a_h^{(j)} - a_h^{(i)}), \quad 1 \leq h \leq H, \quad n_i \leq n_j. \tag{14}$$

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 ($n_j - n_i$).

Fig. 3 shows three successive spectral ramps. Representing an amplitude spectrum evolution 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 [16], the break points 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.

3.4 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.

3.4.1 Spectral Ramp Interpolation Using Original Spectra

Consider Eq. (13), which defines an interpolated spectrum $S^{(ij)}(n)$ on each sample n within the ramp $S^{(ij)}$. For the purpose of measuring the error on the spectral ramp $S^{(ij)}$, we need to compare the successive computed spectra $S^{(ij)}(n)$, $n_i \leq n < n_j$, to the corresponding sequence of DFT spectra $S^{(l)}$ with $i \leq l < j$ (since $n_i = iP$ and $n_j = jP$). Thus we need to sample the computed spectra at the same rate as the DFT spectra,

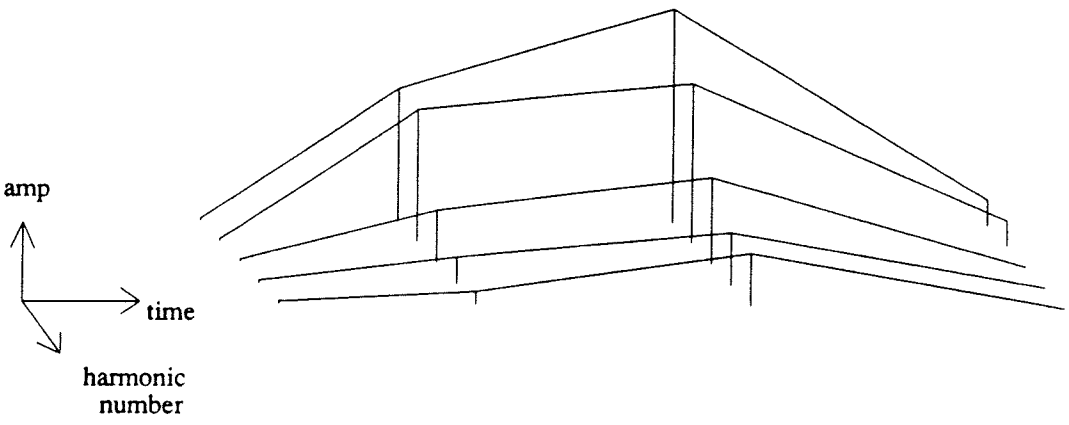

Fig. 3. Three spectral ramps.

that is, 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 $n_l = lP$, that is, $S^{\langle ij\rangle}\langle l\rangle = S^{\langle ij\rangle}(n_l)$. Given this notation, Eq. (14) can be rewritten,

$$a_h^{\langle ij\rangle}\langle l\rangle = a_h^{\langle i\rangle} + \frac{n_l - n_i}{n_j - n_i}(a_h^{\langle j\rangle} - a_h^{\langle i\rangle}),$$

$$i \leq l < j . \quad (15)$$

Comparing the amplitudes of the harmonics of spectrum $S^{\langle l\rangle}$ with values given by Eq. (15) produces an error $E_{\langle l\rangle}^{\langle ij\rangle}$, which is defined as the mean square error on the H harmonic amplitudes,[5]

$$E_{\langle l\rangle}^{\langle ij\rangle} = \sum_{h=1}^{H} (a_h^{\langle ij\rangle}\langle l\rangle - a_h^{\langle l\rangle})^2 . \quad (16)$$

Using the following notation:

$$\begin{aligned} \Delta a_h^{ij} &= a_h^{\langle j\rangle} - a_h^{\langle i\rangle} \\ \Delta a_h^{il} &= a_h^{\langle l\rangle} - a_h^{\langle i\rangle} \\ \Delta n^{ij} &= n_j - n_i \\ \Delta n^{il} &= n_l - n_i \end{aligned} \quad (17)$$

Eq. (15) becomes

$$a_h^{\langle ij\rangle}\langle l\rangle = a_h^{\langle i\rangle} + \frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{ij} . \quad (18)$$

From Eqs. (18) and (16) we deduce

$$\begin{aligned} E_{\langle l\rangle}^{\langle ij\rangle} &= \sum_{h=1}^{H} \left(\frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{ij} + a_h^{\langle i\rangle} - a_h^{\langle l\rangle}\right)^2 \\ &= \sum_{h=1}^{H} \left(\frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{ij} - \Delta a_h^{il}\right)^2 . \end{aligned} \quad (19)$$

The global erorr $E^{\langle ij\rangle}$ within the spectral ramp $S^{\langle ij\rangle}$ is defined as the sum of the errors on the individual spectra,[6]

$$E^{\langle ij\rangle} = \sum_{l=i+1}^{j-1} E_{\langle l\rangle}^{\langle ij\rangle} . \quad (20)$$

If the error $E^{\langle ij\rangle}$ is less than the tolerated threshold $E_{\max}$, we extend the spectral ramp to the next period $(j + 1)$ and compute the new error $E^{\langle i,j+1\rangle}$ using Eq. (20), $E_{\langle l\rangle}^{\langle i,j+1\rangle}$ being computed with Eq. (19). Otherwise we store the data defining the previous spectral ramp $S^{\langle i,j-1\rangle}$, and we compute the next ramp starting at spectrum $(n_{j-1}, S^{\langle j-1\rangle})$.

3.4.2 Spectral Ramp Interpolation Using Computed Spectra

The linear regression algorithm is a way of fitting piecewise linear functions to a set of points [17]. We use a variant of linear regression called *anchored regression* [18]. 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 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 denoted by $S'^{\langle i\rangle}$. The prime is used to differentiate the anchors from the DFT spectra. The anchored regression algorithm works as follows. We start with an anchor $(n_i, S'^{\langle i\rangle})$. For the first anchor we take $(n_0, S^{\langle 0\rangle})$.[7] For each successive consecutive DFT spectrum $(n_j, S^{\langle j\rangle})$, $j > i$, we compute H lines. The hth line goes through its anchor $(n_i, a_h'\langle i\rangle)$ and comes closest (in the least-squares sense) to the set of points $(n_{i+1}, a_h\langle i + 1\rangle, \ldots, (n_j, a_h\langle j\rangle)$. 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 $(n_{j+1}, S^{\langle j+1\rangle})$ is examined. Otherwise the spectral ramp $S'^{\langle i,j-1\rangle}$ is used. In this case the endpoint of the spectral ramp is (n_{j-1}, S'^{j-1}). The H coordinates of $S'^{\langle j-1\rangle}$, $a_h'\langle j - 1\rangle$, for $1 \leq h \leq H$, are computed using the slopes of the H best lines. The endpoint $(n_{j-1}, S'^{\langle j-1\rangle})$ 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 $S'^{\langle ij\rangle}$ must go through the coordinates $(n_i, a_h'\langle i\rangle)$, each line is defined by an equation of the form

$$a'^{\langle ij\rangle}_h\langle l\rangle = m_h^{\langle ij\rangle}(n_l - n_i) + a_h'\langle i\rangle,$$

$$1 \leq h \leq H, \quad i \leq l \leq j \quad (21)$$

where $m_h^{\langle ij\rangle}$ is the slope of the hth line of the ramp. Defining $E_h^{\langle ij\rangle}$ to be the sum of the squared errors of each harmonic within the ramp $S'^{\langle ij\rangle}$, we have

$$E_h^{\langle ij\rangle} = \sum_{l=i+1}^{j} [a_h\langle l\rangle - (m_h^{\langle ij\rangle}(n_l - n_i) + a_h\langle l\rangle)]^2 . \quad (22)$$

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

[6] We may start the sum from $i + 1$ since as we use $S^{\langle i\rangle}$ as the initial spectrum in the spectral ramp, we have $E_{\langle i\rangle}^{\langle ij\rangle} = 0$.

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

The total error on the spectral ramp $E^{\langle ij \rangle}$ is the sum of the errors on each harmonic,

$$E^{\langle ij \rangle} = \sum_{h=1}^{H} E_h^{\langle ij \rangle} . \quad (23)$$

By using the notation previously defined [Eq. (17)] we can write

$$E_h^{\langle ij \rangle} = \sum_{l=i+1}^{j} (\Delta a_h^{il} - m_h^{\langle ij \rangle} \Delta n^{il})^2 \quad (24)$$

$$E^{\langle ij \rangle} = \sum_{h=1}^{H} \left[\sum_{l=i+1}^{j} (\Delta a_h^{il} - m_h^{\langle ij \rangle} \Delta n^{il})^2 \right] . \quad (25)$$

Differentiating the error with respect to $m_h^{\langle ij \rangle}$, we obtain

$$\frac{\partial E^{\langle ij \rangle}}{\partial m_h^{\langle ij \rangle}} = -2 \sum_{l=i+1}^{j} (\Delta a_h^{il} - m_h^{\langle ij \rangle} \Delta n^{il}) \Delta n^{il} . \quad (26)$$

Setting each derivative $\partial E^{\langle ij \rangle} / \partial m_h^{\langle ij \rangle} = 0$ gives the slope of each line,

$$m_h^{\langle ij \rangle} = \frac{\sum_{l=i+1}^{j} \Delta n^{il} \Delta a_h^{il}}{\sum_{l=i+1}^{j} (\Delta n^{il})^2} . \quad (27)$$

As we mentioned before, if the total error $E^{\langle ij \rangle}$ is less than the threshold, we add the next spectrum $S^{\langle j+1 \rangle}$ to our set and calculate the error $E^{\langle i,j+1 \rangle}$. Interestingly, we do not have to reevaluate from scratch the sums in Eqs. (25) and (27) when we add the new spectra. Rewriting (24) gives

$$E_h^{\langle ij \rangle} = \sum_{l=i+1}^{j} (\Delta a_h^{il})^2 - 2 m_h^{\langle ij \rangle} \sum_{l=i+1}^{j} \Delta n^{il} \Delta a_h^{il} + (m_h^{\langle ij \rangle})^2 \sum_{l=i+1}^{j} (\Delta n^{li})^2 . \quad (28)$$

Now each of the sums in Eqs. (27) and (28) may be computed incrementally:

$$\sigma_{nn}^{\langle ii \rangle} = \sigma_{aa}^{\langle ii \rangle}(h) = \sigma_{na}^{\langle ii \rangle}(h) = 0$$

$$\sigma_{nn}^{\langle il \rangle} = \sigma_{nn}^{\langle i,l-1 \rangle} + (\Delta n^{l-1,l})^2$$

$$\sigma_{aa}^{\langle il \rangle}(h) = \sigma_{aa}^{\langle i,l-1 \rangle}(h) + (\Delta a_h^{l-1,l})^2$$

$$\sigma_{na}^{\langle il \rangle}(h) = \sigma_{na}^{\langle i,l-1 \rangle}(h) + \Delta a_h^{l-1,l} \Delta n^{l-1,l},$$

$$i < l \leq j, \quad 1 \leq h \leq H . \quad (29)$$

Now $m_h^{\langle ij \rangle}$ [Eq. (27)] and $E_h^{\langle ij \rangle}$ [Eq. (28)] can be computed as

$$m_h^{\langle ij \rangle} = \frac{\sigma_{na}^{\langle ij \rangle}(h)}{\sigma_{na}^{\langle ij \rangle}(h)}$$

$$E_h^{\langle ij \rangle} = \sigma_{aa}^{\langle ij \rangle}(h) - 2 m_h^{\langle ij \rangle} \sigma_{na}^{\langle ij \rangle}(h) + m_h^{\langle ij \rangle 2} \sigma_{nn}^{\langle ij \rangle} . \quad (30)$$

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 $E^{\langle ii \rangle}, E^{\langle i,i+1 \rangle}, \ldots, E^{\langle ij \rangle}$ with the same effort as it takes to just compute $E^{\langle ij \rangle}$.

3.4.3 First Results

After computing the spectral ramps (using either method), we resynthesize the tone by evaluating Eqs. (4) and (2) 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 used in the Bradford musical instrument simulator [19]. After two waveforms defining a spectral ramp have been loaded in the wave tables, the wave tables are read, scaled by two opposite linear ramps [Eq. (13)], 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 Fig. 4 shows a sampled trombone tone.[8] Beneath the tone are the spectra that resulted from the DFT analysis of the tone. Under those are the spectra selected by the data reduction algorithm. The synthesized tone is shown last.

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 $d(n)$ [Eq. (12)] 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.

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

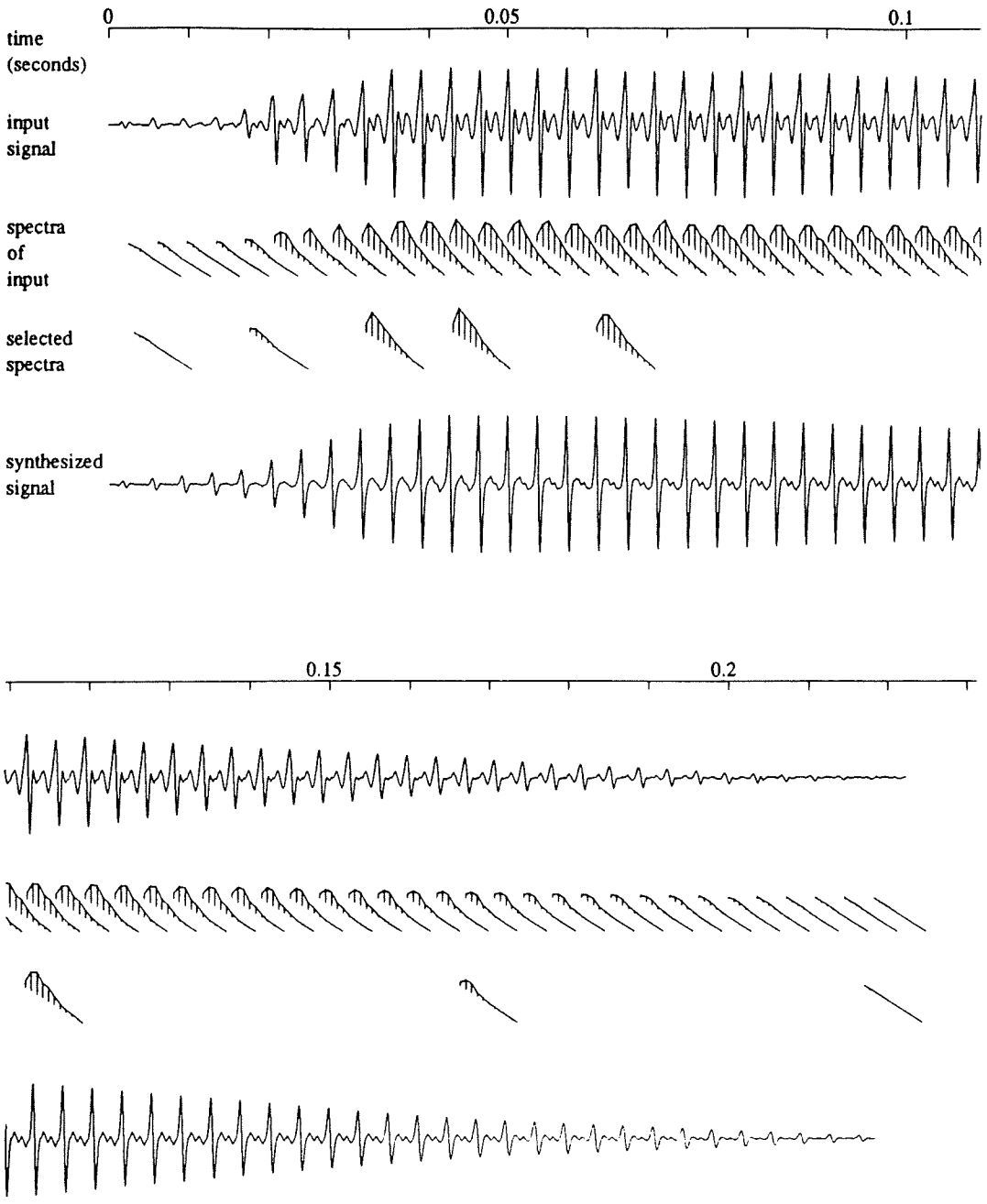

Fig. 4. Synthesis by time-linear spectral interpolation.

3.5 Data-Reduction Using the Nonlinear Spectral Interpolation Representation

In this section we consider that the paths connecting the harmonics [Eq. (11)] $c(n)$ 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 $S^{\langle i \rangle}$ and $S^{\langle j \rangle}$ 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, that is, we try the spectral path that links together the spectra $S^{\langle i \rangle}$ and $S^{\langle j-1 \rangle}$, and so on.

Within the spectral path $S^{\langle ij \rangle}$ the harmonics of the interpolated spectrum $S^{\langle ij \rangle}_{\langle l \rangle}$ at period l are computed using Eq. (11)

$$a^{\langle ij \rangle}_{\langle h \rangle}\langle l \rangle = c(n_l)a_h\langle i \rangle + d(n_l)a_h\langle j \rangle \quad \text{with } n_l = lP, \quad i \leq l < j \ . \tag{31}$$

Using the notation

$$c_l = c(n_l) \qquad d_l = d(n_l)$$

Eq. (31) can be written as

$$a^{\langle ij \rangle}_{\langle h \rangle}\langle l \rangle = c_l a_h\langle i \rangle + d_l a_h\langle j \rangle \tag{32}$$

or, equivalently,

$$S^{\langle ij \rangle}_{\langle l \rangle} = c_l S^{\langle i \rangle} + d_l S^{\langle j \rangle} \ . \tag{33}$$

The error on spectrum $S^{\langle ij \rangle}_{\langle l \rangle}$ within the ramp $S^{\langle ij \rangle}$ is defined as the mean square error on the harmonics,[9]

$$E^{\langle ij \rangle}_{\langle l \rangle} = \sum_{h=1}^{H} [a_h\langle l \rangle - (c_l a_h\langle i \rangle + d_l a_h\langle j \rangle)]^2 \ . \tag{34}$$

The error is minimized by setting the two partial derivatives $\partial E^{\langle ij \rangle}_{\langle l \rangle}/\partial c_l$ and $\partial E^{\langle ij \rangle}_{\langle l \rangle}/\partial d_l$ to zero,

$$\frac{\partial E^{\langle ij \rangle}_{\langle l \rangle}}{\partial c_l} = 2c_l \sum_{h=1}^{H} a_h\langle i \rangle^2 + 2d_l \sum_{h=1}^{H} a_h\langle i \rangle a_h\langle j \rangle - 2 \sum_{h=1}^{H} a_h\langle l \rangle a_h\langle i \rangle = 0$$

$$\frac{\partial E^{\langle ij \rangle}_{\langle l \rangle}}{\partial d_l} = 2d_l \sum_{h=1}^{H} a_h\langle j \rangle^2 + 2c_l \sum_{h=1}^{H} a_h\langle i \rangle a_h\langle j \rangle - 2 \sum_{h=1}^{H} a_h\langle l \rangle a_h\langle j \rangle = 0 \ .$$

This is a two-equation linear system with unknowns c_l and d_l

$$c_l \sum_{h=1}^{H} a_h\langle i \rangle^2 + d_l \sum_{h=1}^{H} a_h\langle i \rangle a_h\langle j \rangle = \sum_{h=1}^{H} a_h\langle l \rangle a_h\langle i \rangle$$

$$c_l \sum_{h=1}^{H} a_h\langle i \rangle a_h\langle j \rangle + d_l \sum_{h=1}^{H} a_h\langle j \rangle^2 = \sum_{h=1}^{H} a_h\langle l \rangle a_h\langle j \rangle \ . \tag{35}$$

The existence of a solution (c_l, d_l) to the system (35) depends on the value of its determinant and on the values of the constant terms of the system. Consider the initial equation [Eq. (33)]. If the two vectors $S^{\langle i \rangle}$ and $S^{\langle j \rangle}$ are linearly independent (in a vector space of dimension H), it is possible to find a unique solution (c_l, d_l) that minimizes the error. Otherwise the problem remains underdetermined or impossible.[10]

The nonlinear interpolation model has been tried on a number of tones. In practice, there are two kinds of situations for which Eqs. (35) have no unique solution:

1) One of the vectors $S^{\langle i \rangle}$ or $S^{\langle j \rangle}$ is zero or close to zero. This happens usually when one of the spectra is located in a low-energy segment of the signal (for example, at the beginning or at the end of the note).

2) The vectors are not zero but are linearly dependent or nearly dependent. This happens usually in the sustain part of the envelope, when the tone is played with a constant loudness or when the amplitudes of the harmonics are varying strictly linearly (which is rare on natural sounds).[11]

We have investigated several methods for dealing with such cases. In the first method, Eq. (33) is replaced by a one-vector decomposition by, for example, assuming $d_l = 0$,

$$S^{\langle ij \rangle}_{\langle l \rangle} = c_l S^{\langle i \rangle} \tag{36}$$

c_l being a real number. Expressing Eq. (36) in terms of the harmonic amplitudes yields

$$a^{\langle ij \rangle}_{\langle h \rangle}\langle l \rangle = c_l a_h\langle i \rangle, \qquad 1 \leq h \leq H \ . \tag{37}$$

The error on spectrum $S^{\langle ij \rangle}_{\langle l \rangle}$ is now

$$E^{\langle ij \rangle}_{\langle l \rangle} = \sum_{h=1}^{H} (a^{\langle ij \rangle}_{h}\langle l \rangle - a_h\langle l \rangle)^2 = \sum_{h=1}^{H} (c_l a_h\langle i \rangle - a_h\langle l \rangle)^2 \ . \tag{38}$$

Setting the derivative of the error with respect to c_l to

[9] Here again we omit the averaging factor $1/H$.

[10] There is an infinity of solutions when the constant terms are zero, otherwise there are no solutions.

[11] When the two vectors $S^{\langle i \rangle}$ and $S^{\langle j \rangle}$ as well as $S^{\langle l \rangle}$ are close to zero but not strictly zero, we can find a solution.

zero we obtain

$$c_l = \frac{\sum_{h=1}^{H} a_h\langle i\rangle a_h\langle l\rangle}{\sum_{h=1}^{H} a_h\langle i\rangle^2} . \quad (39)$$

The error on each spectrum within the spectral path can be computed by Eq. (38) and the error within the spectral ramp is taken as the sum of the errors on the individual spectra. If the global error exceeds the threshold, a shorter spectral path will be tried.

This method is straightforward. Unfortunately it is not reliable. The main problem is an occasionally perceptible discontinuity between the two spectra $S^{\langle ij\rangle}_{\langle j-1\rangle} = c_{j-1}S^{\langle i\rangle}$ and $S^{\langle j\rangle}$. If it was the case that $S^{\langle i\rangle}$ and $S^{\langle j\rangle}$ were truly dependent, there would be no discontinuity. However, our test for dependence checks only that the determinant is very small, not that it is exactly zero, so it is usually the case that $S^{\langle i\rangle}$ and $S^{\langle j\rangle}$ are not strictly dependent. This difference can be occasionally perceived as a soft click in the signal. This is exactly the same problem one gets when switching waveforms without interpolation.

The second method uses a linear interpolation to avoid the discontinuity when we abruptly change spectra,

$$S^{\langle ij\rangle}_{\langle l\rangle} = c_l\left(S^{\langle i\rangle} + \frac{n_l - n_i}{n_j - n_i}(S^{\langle j\rangle} - S^{\langle i\rangle})\right) \quad (40)$$

so that $d_l = c_l(n_l - n_i)/(n_j - n_i)$. The amplitudes of the harmonics in spectrum $S^{\langle ij\rangle}_{\langle l\rangle}$ are

$$a_h^{\langle ij\rangle}\langle l\rangle = c_l\left(a_h\langle i\rangle + \frac{n_l - n_i}{n_j - n_i}(a_h\langle j\rangle - a_h\langle i\rangle)\right), \quad 1 \leq h \leq H . \quad (41)$$

The error $E^{\langle ij\rangle}_{\langle l\rangle}$ is expressed as

$$E^{\langle ij\rangle}_{\langle l\rangle} = \sum_{h=1}^{H}\left[a_h\langle l\rangle - c_l\left(a_h\langle i\rangle\frac{n_j - n_l}{n_j - n_i} + a_h\langle j\rangle\frac{n_l - n_i}{n_j - n_i}\right)\right]^2 . \quad (42)$$

By minimizing the error (42) with respect to c_l we obtain

$$c_l = \frac{\sum_{h=1}^{H} a_h\langle l\rangle \sum_{h=1}^{H}\left(a_h\langle i\rangle\frac{n_j - n_l}{n_j - n_i} + a_h\langle j\rangle\frac{n_l - n_i}{n_j - n_i}\right)}{\sum_{h=1}^{H}\left(a_h\langle i\rangle\frac{n_j - n_l}{n_j - n_i} + a_h\langle j\rangle\frac{n_l - n_i}{n_j - n_i}\right)^2} . \quad (43)$$

The resulting error is obtained by replacing c_l by its value in Eq. (42).

An alternative formulation would be to take $d_l = (n_l - n_i)/(n_j - n_i)$ so that

$$S^{\langle ij\rangle}_{\langle l\rangle} = c_l S^{\langle i\rangle} + \frac{n_l - n_i}{n_j - n_i} S^{\langle j\rangle} \quad (44)$$

$$a_h^{\langle ij\rangle}\langle l\rangle = c_l a_h\langle i\rangle + \frac{n_l - n_i}{n_j - n_i} a_h\langle j\rangle . \quad (45)$$

This yields the optimal c_l,

$$c_l = \frac{\sum_{h=1}^{H}\left(a_h\langle i\rangle a_h\langle l\rangle - \frac{n_l - n_i}{n_j - n_i} a_h\langle j\rangle a_h\langle l\rangle\right)}{\sum_{h=1} a_h\langle i\rangle^2} . \quad (46)$$

These last two variants of nonlinear interpolation assure the continuity of the output signal, while allowing a high spectral reduction rate.

By applying the nonlinear interpolation analysis we have been able to reduce greatly the number of waveforms needed to resynthesize the tone accurately. The reduction is greatest during the sustain portion of the tone. However, we now need many points (two numbers per period) to represent the scaling functions c_l and d_l. These scaling functions are usually well behaved, so we are able to approximate them with piecewise linear functions containing a small number of break points. This we do after we have computed the spectral paths representing the entire tone. We use the anchored regression algorithm of [18] to perform the fitting automatically.

We do not have an incremental algorithm for computing spectral paths as we had for computing spectral ramps (Sec. 3.4.1). Thus the effort expended in computing the optimal spectral path between $S^{\langle i\rangle}$ and $S^{\langle j\rangle}$ does not help at all in computing the optimal path between $S^{\langle i\rangle}$ and $S^{\langle j+1\rangle}$, or between $S^{\langle i\rangle}$ and $S^{\langle j-1\rangle}$. Our nonlinear interpolation algorithm which iterates until the error threshold is crossed (as do the time-linear algorithms of Sec. 3.4.1) is quite time consuming. We could reduce this time by doing binary search, or better yet by developing some kind of incremental algorithm. We have done neither of these, but have instead experimented with a fixed spectral path length, as we now describe.

To speed up the analysis process we have the option of using an arbitrary fixed interpolation interval, that

is, we approximate the tone with a list of spectral paths of identical length. On a given tone, we run this algorithm several times, varying the spectral path length. We then listen to the resulting resynthesized tones and choose the one with the longest spectral paths (fewest spectra) which sounds indistinguishable from the original. Fig. 5 shows the same input tone as Fig. 4, but synthesized via the nonlinear interpolation method.

3.6 Results

The nonlinear interpolation analysis using fixed-length spectral paths (Sec. 3.5) and the linear interpolation analysis (Sec. 3.4) are both very efficient. The linear model is more efficient, partly due to the fact that the nonlinear model requires an additional smoothing of the synthesis input parameters. The nonlinear analysis generally results in an improved data reduction rate.

Table 1 shows results of applying linear and nonlinear interpolation to several tones. All the tones shown in this table were digitized and resynthesized at a 16-kHz sample rate. The "duration" column gives the duration of the tone in seconds, P is the period of the tone in number of samples, N_P is the total number of periods in the tone (and thus the total number of DFT spectra), H is the number of harmonics in each spectrum, "method" is the algorithm (linear or nonlinear) used for the analysis, Q is the number of spectra used in the synthesis (constant for a given tone), Q/s is the number of spectra per second sent to the synthesizer, R is the spectral reduction rate, and Nbytes/s is the number of 8-bit bytes per second sent to the synthesizer. The latter

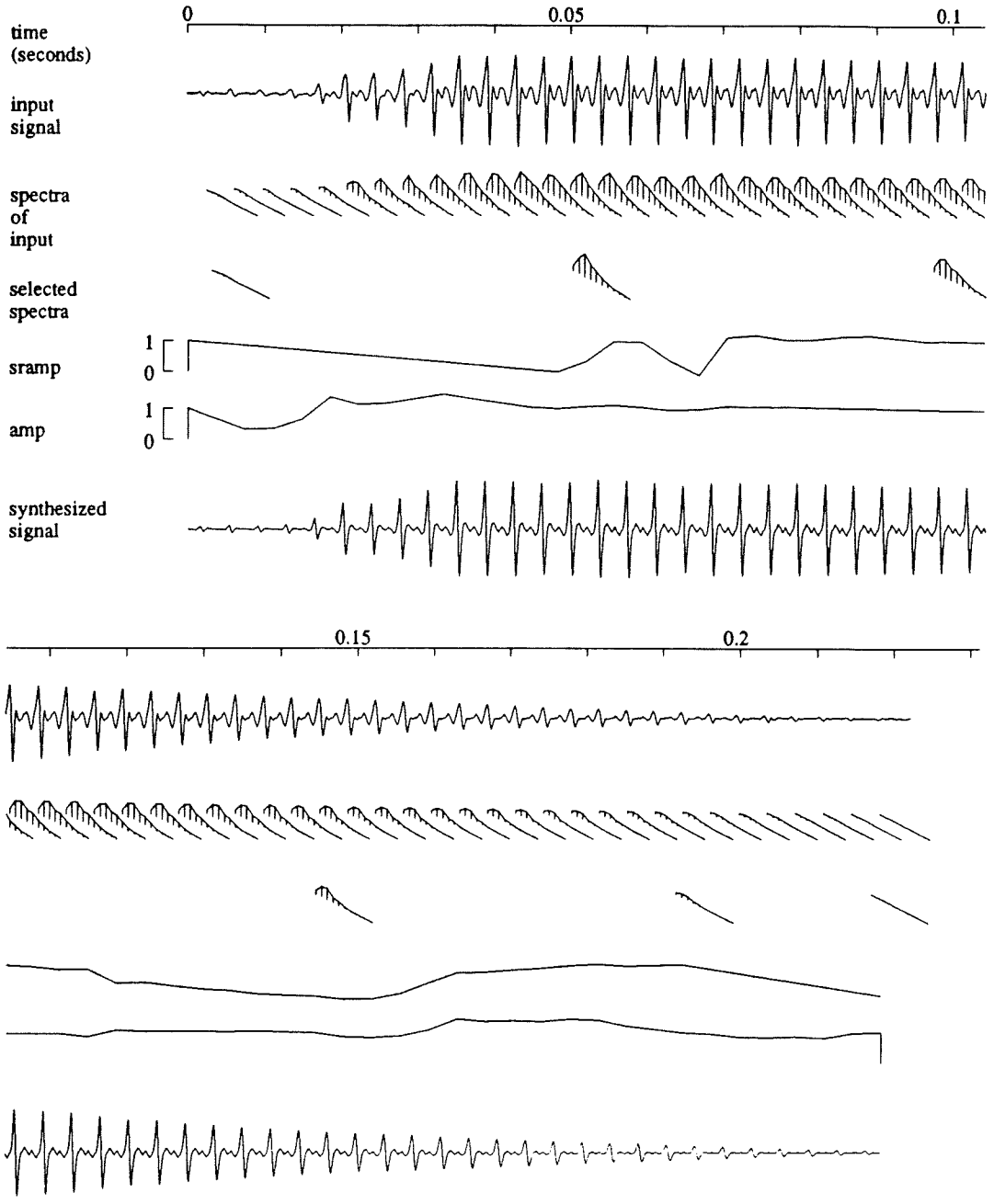

Fig. 5. Synthesis by nonlinear spectral interpolation. C = amp · sramp; d = amp (1 − sramp).

three columns are used to characterize the efficiency of the analysis/synthesis process:

1) Q/s is the number of spectra per second sent to the synthesizer, Q/duration.

2) R is the spectral reduction rate, Q/N_P, in percent, and thus compares the number of spectra in the synthesized tone to the number of periods in the original tone.

3) Nbytes/s is the number of 8-bit bytes per second sent to the synthesizer. It is equal to the number of spectra per second, Q/duration, multiplied by the number of harmonics H (that is, the amplitude of each harmonic is coded using one byte as in [19]). For nonlinear interpolation, the number of break points per second of $c(n)$ and $d(n)$ are added assuming 2 bytes per break point.

Each of the five tones shown in Table 1 have been synthesized via both methods. For a given tone and method, the tone has been resynthesized using a number of different threshold values. Of these, the tone selected to be shown in the table is the tone synthesized using the smallest number of spectra that is judged to be indistinguishable from the original tone. Thus the numbers in Table 1 are subjective, but we believe they provide an accurate comparison between our two synthesis methods, and between our synthesis methods and other analysis/synthesis techniques.

In Table 1 we did not mention which linear method was used to compute the spectral ramps (Sec. 3.4.1 or 3.4.2) as the two methods perform almost identically.

Since the result of spectral interpolation can be computed exactly using additive synthesis, it is interesting to compare the two methods in terms of efficiency. We use the following parameters: sample rate r, table size N, number of harmonics h, number of waveforms per second w, and number of operations per second o.

For additive synthesis, we have one oscillator per harmonic (h oscillators) and each sample requires one multiplication for the amplitude, one addition to increment the phase, and one addition to sum the oscillator output into an accumulator. The total number of operations per second is

$$o = 3rh \ .$$

This figure ignores the computation of amplitude ramp functions which could contribute as much as another addition and a comparison to each sample of each oscillator. In practice, amplitude ramps are often computed at low sample rates to reduce the computation requirement.

For spectral interpolation synthesis, a waveform computation requires a multiply and two adds for each harmonic and each waveform sample for a total of $3hN$ operations. To increment the phase and interpolate between two tables requires three additions and two multiplies, or five operations per sample. Multiplying by the waveform rate w and sample rate r, respectively, gives

$$o = 3hNw + 5r \ .$$

The ratio of spectral interpolation to additive synthesis computation is

$$\frac{5r + 3hNw}{3hr} = \frac{5}{3h} + \frac{wN}{r} \ .$$

The $5/3h$ term represents one interpolating oscillator versus h sine-wave oscillators. The term wN/r represents the cost of summing harmonics into a wave table versus computing harmonics at the sample rate.

Using actual data from our experiments, we find that the computation cost for spectral synthesis is dominated by the cost of computing waveforms. Thus the most critical factors are the size of the tables and the number of tables computed per second. The relative advantage of spectral interpolation over additive synthesis increases with the sample rate. For a table size of 512 samples and a sample rate of 16 kHz, the ratio varies from 0.30 to 0.63 in typical data. If the sample rate were increased to 48 kHz, the ratio would vary from 0.13 to 0.24. This represents a substantial reduction in the number of operations required by spectral interpolation synthesis with respect to summation synthesis, even if we ignore the cost of extra amplitude ramps and additional control information also required by additive synthesis.

The tones compared are those that gave the best results. These tones belong to the woodwind family. The resynthesis of brass instruments using similar data reduction rates produced acceptable but not identical tones when compared to the originals. In particular, the attacks of the trumpet and the trombone tones did not sound perfectly natural. In fact, increasing the number of spectra did not achieve the desired matching between

Table 1.

Name	Duration	P	N_P	H	Method	Q	Q/s	R	Nbytes/s
Bassoon 1	1.44	204	111	40	Linear	16	11.1	14%	444
Bassoon 1					Nonlinear	12	8.3	11%	388
Bassoon 2	2.12	136	249	40	Linear	19	9.0	8%	359
Bassoon 2					Nonlinear	14	6.6	6%	323
Bassoon 3	1.06	81	211	30	Linear	18	17.0	9%	509
Bassoon 3					Nonlinear	11	10.4	5%	400
Clarinet 1	1.25	103	194	40	Linear	23	18.4	12%	736
Clarinet 1					Nonlinear	20	18.4	10%	704
Clarinet 2	3.77	91	663	25	Linear	21	5.6	3%	139
Clarinet 2					Nonlinear	18	4.8	3%	175

the synthesized and the original attack. We believe that this is due to the fact that our analysis/synthesis method does not handle inharmonicity in general, and brass tones often show inharmonic properties in the attack. When we apply the pitch synchronous DFT on a nonharmonic attack, we get erroneous spectral measures; furthermore, even if we resort to a more general spectral analysis model (such as the phase vocoder), it is still difficult to use spectral interpolation synthesis to reproduce inharmonicities (see Sec. 2). In order to keep the advantages of the spectral interpolation model while achieving more natural attacks, we have investigated a technique in which a sampled attack is spliced onto a synthesized sustain, as we describe in the next section. An evaluation of this hybrid technique is given at the end of that section.

3.7 Combination of Sampling and Spectral Interpolation Synthesis

We now consider the problem of reproducing a tone by connecting the attack portion of the tone to a synthesized sustain portion obtained by analyzing the tone using one of our algorithms described previously.[12] We first examine methods of deciding how much of the original signal to use. We then discuss the techniques we tried for connecting the sampled and synthesized portions.

For data reduction it is desirable to use as small a sampled attack portion as possible.[13] On the other hand, we need enough of the attack to create a convincing reproduction of the tone. Since our synthesis method only reproduces harmonic sounds, the attack sample should include the significant nonperiodic or inharmonic portions of the tone. For our technique to be useful it is assumed that these inharmonicities occur at the start of the tone and are of short duration. As one might expect, the two methods we use for pitch detection (optimum comb and peak detection) show erratic behavior during the inharmonic portions of the attack. By observing where the pitch detectors settle down, we attempt to get an idea of where the attack inharmonicities finish and the harmonic sustain begins. In actuality this method is not totally reliable (at least we have not worked at making it so), so we are occasionally forced to choose the attack endpoint manually by examining a plot of the signal.

Connecting a sampled attack to a synthesized sustain requires that for each harmonic the phase of the sampled harmonic matches the phase of the corresponding synthesized harmonic at the transition point. If some corresponding harmonic phases are not equal, we may hear a click in the signal due to the instantaneous phase shifts. If the transition occurs at a zero crossing in the signal, the click is attenuated, but often still perceivable. We have tried two techniques for achieving smooth transitions, *phase interpolation* and *simple phase matching*.

In the phase interpolation technique we attempt to gradually shift the phases of the harmonics in the sampled attack to be zero or π[14]. To this end, we divided the attack portion into two segments. The first is used unchanged; the second is processed so that by the end of it the harmonic phases are zero or π.[15] We have to find a way of shifting the initial phases of the harmonics of the second segment to zero so that the frequency shift resulting from the phase shift is imperceptible. We tried to use a phase interpolation algorithm, based on [21], where the phases are progressively interpolated (using cubic polynomials) from their original values to zero or π. In order to reduce an eventual perceptual effect, we decrease the phase shift per sample by using a long interpolation time interval, from 5 to 10 periods or more, if necessary. Also, to reduce the global frequency shift, we tried to use opposite directions for each harmonic. In any case, the phase interpolation algorithm has been successful on those few cases where the phase shifts to be applied were very small. Unfortunately, for real signals such as trumpet and trombone tones, the shifts that we had to apply were quite large. In most cases, the phase interpolation was always perceived as a frequency shift whose magnitude was not acceptable. Although this is a negative result, we found it surprising: in general the relative phases of the harmonics of a signal cannot be arbitrarily rearranged in an imperceptible manner, even over many periods. Consider adjusting the relative phase of the second harmonic. In the worst case the harmonic will be π rad out of phase, but this represents only $\pi/2$ rad of the fundamental. If an adjustment is made over 10 periods, we are shifting the second harmonic by at most $\pi/20$ rad per period. Higher harmonics will have even less phase shift per period, and yet the shift is clearly perceptible.

Thus we decided to compute the first waveform of the synthesis using the phases measured at the transition. This is the simple phase matching technique. As we can interpolate only between spectra having identical

[12] Here we are using the term "sustain" to refer to the part of a tone after the attack. It includes what is normally thought of as the sustain portion as well as the release portion of the tone.

[13] There are other reasons to want a short attack sample. In particular, as the control of a sampling synthesizer is limited compared to that of the interpolation synthesizer, we desire the duration of the sampled portion to be as small as possible.

[14] We had an ulterior motive for trying to do this. We wish to use the Bradford musical instrument simulator [20] hardware for our synthesis. This hardware currently requires the phases of the harmonics in the wave table to be zero or π. A number of benefits come as a consequence of this restriction: the transition between successive wave table lookups is continuous, the update rate of the amplitude scaling factors in the oscillator can be low, only a small number of bits are needed to code the amplitude factors, and very cheap multipliers (gate arrays) can be used. We would have liked to maintain these advantages.

[15] Ideally, the point where the attack is broken should be one where the harmonic phases are as near as possible to their final values, zero or π. We wrote programs which searched for such points, but in practice it was never possible to find one where more than a few of the harmonics were close to their respective targets.

phases, all the waveforms used during the synthesis are computed with those phases. By doing this, the phase match at the transition is faultless. In addition, computing the waveforms with the phases found at the end of the attack did not alter the quality of the synthesized signal.

Using the simple phase matching technique, we have obtained very high quality reproductions of brass instruments. Fig. 6 shows the same input tone as Figs. 4 and 5, but synthesized using a sampled attack. The arrow points to the transition between sampled sound and synthesized sound.

Table 2 depicts the data reduction rates obtained by combining sampling and spectral interpolation on several tones. It uses the following parameters: "duration" is the total duration of the tone in seconds, *s*rate is the

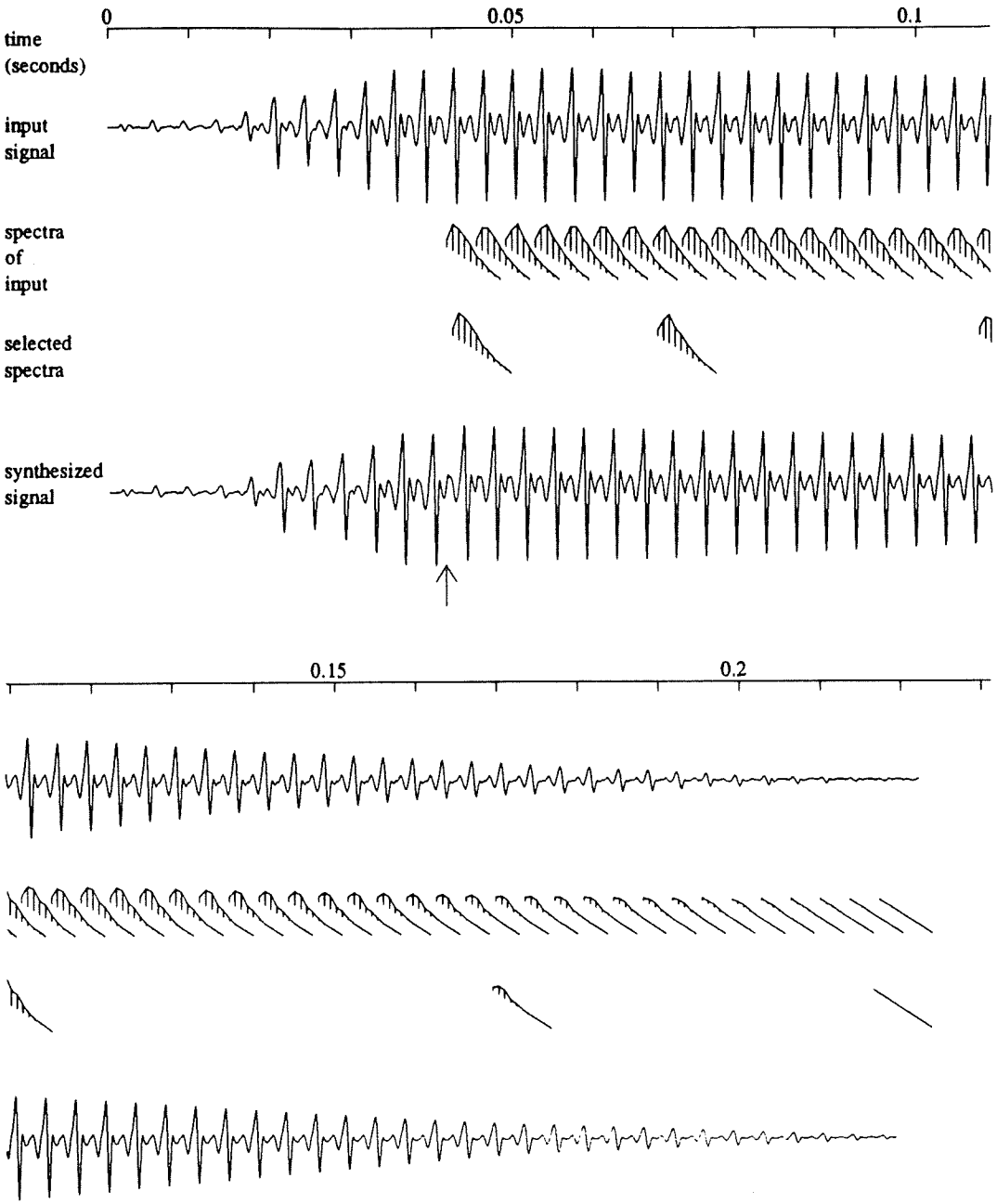

Fig. 6. Synthesis by spectral interpolation with sampled attack.

Table 2.

Name	Duration	*s*rate	*P*	N_P	*H*	Method	*Q*	*Q*/*S*	*R*	Attack	*N*bytes/*s*
Trombone 1	0.39	13513.5	58	81	25	Linear	5	14.2	6%	500	320
Trombone 2	0.96	13513.5	58	210	25	Linear	5	5.4	2%	500	130
Trumpet 1	0.55	13201	30	211	6	Linear	5	10.5	2%	1000	54
Trumpet 2	0.99	13201	30	389	6	Linear	7	7.7	2%	1000	42
Saxophone 1	4.82	16604	46	1666	20	Nonlinear	63	13.2	4%	500	300

sample rate in kilohertz, P is the period of the tone in number of samples, N_P is the total number of periods in the tone, H is the number of harmonics, Q is the number of spectra sent to the synthesizer, "attack" is the number of samples copied directly from the original tone, Nbytes/s is the number of 8-bit bytes per second sent to the synthesizer (excluding the attack samples), and "method" is the linear or nonlinear interpolation algorithm.

The hybrid technique, sampling and spectral interpolation, can be applied without restriction to any kind of sound whose sustain is harmonic, which is the case for most orchestral instruments.

4 RELATED WORK

We have presented and discussed the technique of waveform interpolation for the analysis and resynthesis acoustic instruments. We use a succession of wave tables that are dynamically mixed to reproduce analyzed spectral evolutions. Until now, waveform interpolation has been used in a number of other contexts by different people. Some of these past applications are similar to ours, some not. We now review this related work.

4.1 Mixing Waveforms to Generate Dynamic Sounds

The idea of mixing multiple sequences of samples to generate sounds whose spectra change over time is not new. Several commercial synthesizers have implemented this technique. The digital instrument built by Matsushita [10] reproduces acoustic instruments with a similar oscillator as that in Fig. 1. In this instrument, the waveforms are extracted manually from real tones. In fact, to avoid phase problems Matsushita [10] advocates using the Fourier transform. The Matsushita digital instrument uses the same synthesis algorithm as we do. However, the work differs from ours as it does not use any automatic analysis.

The Prophet VS [22], [23] from Sequential Circuits uses "digital vector synthesis." Each voice is the result of simultaneously mixing four wave tables. Unlike our oscillator, the wave table contents of the VS are fixed over the course of a note. The four-way interpolation is dynamically altered as a function of envelope generators, keyboard velocity, joystick position, as well as other factors.

The Keytex CTS-2000 is a crosstable sampled synthesizer [24]. It uses the technique of waveform interpolation. A voice is the sum of two oscillators, each with independent wave tables, frequencies, and amplitudes. Each oscillator can generate an interpolated succession of exactly three different complex wave tables. The synthesizer comes with a number of wave tables stored in ROM. Unfortunately there is no way for users to specify their own wave tables.

4.2 Voice Point Interpolation

Voice point interpolation is a technique whereby the user specifies signals or synthesis parameters at a number of points, and interpolation is used to compute the signal at points between the specified points. A point is a location in a space, the dimensions of the space most often being pitch, velocity, key pressure, and other controller input. Voice point interpolation is interesting as it attempts to efficiently model the timbre of an instrument as a function of the control inputs.

The Bradford musical instrument simulator [19], [20] uses voice point interpolation to reproduce timbre variations with pitch and loudness for organ tones, among other instruments. For this two-dimensional space they use two waveform interpolation oscillators. Spectral variation with amplitude is a result of the relative scaling of the two wave tables within each oscillator; spectral variation with frequency is achieved by varying the mix of the output of the two oscillators. It is easy to see how this scheme can be generalized to higher dimensional spaces, three dimensions requiring eight waveforms, four dimensions requiring 16 waveforms, and so on.

4.3 Timbre Interpolation

Research on timbre has also used interpolation of multiple samples or of spectra. Lo [25] uses interpolation of portions of the signal (called break frames) to reproduce timbre. The break frames are normalized in length and recomputed by FFT for phase control. A similar model is applied for interpolating between different timbres [16] to combine timbres dynamically. To interpolate between instruments, Grey [16] uses interpolation of envelope break points defining the amplitude and frequency harmonics.

4.4 Combination of Sampled Sounds with Synthesized Sounds

Smith and Serra [26] independently combined sampled attacks with synthesized tones using the same phase-matching technique as that described in Sec. 3.7. Their synthesis technique was based on summation of sinusoids rather than interpolation. The Roland D-50 [27] seems to be able to combine sampled attacks with synthesized sustains. At present we have no technical information on the method the D-50 uses to ensure smooth transitions.

5 DIRECTIONS FOR FUTURE WORK

We have simplified our analysis by restricting our data to relatively fixed-frequency examples. To relax this restriction, it shoud be possible to resample the input to obtain a fixed frequency, perform the analysis, and then use a time-varying frequency in the table lookup oscillator to reproduce the original frequency. To resample, the time-varying frequency must first be determined by some pitch extraction method. The signal can then be resampled [13] at a sampling rate that varies in proportion to the measured frequency. The effect is to "flatten" the frequency variations and obtain a fixed number of samples per period. The same frequency information can be subjected to linear regression for

data reduction before resynthesis.

We have concentrated on sounds with a highly stable period. It would be interesting to apply the same techniques to bow string and vocal sounds, which might be expected to have some jitter in their periods. Our analysis/synthesis techniques will tend to remove the jitter, resulting in a perceptually altered sound. Can the jitter be introduced into synthesis for more realism? Can this extension to our basic techniques be automated?

Finally, this study is the first stage of a two-stage project to meet the criteria of generality, efficiency, and control. So far we have not discussed the problem of control. Our plan is to develop a characterization of an instrument, be it real or artificial, as a multidimensional space of spectra. Typical dimensions would be amplitude and frequency, which are input parameters to this synthesis model. Other parameters, such as bow position or lip pressure, can be introduced as additional dimensions. Variations in pitch, amplitude, and other parameters give rise to a trajectory in this space, and spectral interpolation can be used to reproduce the corresponding spectral evolution. Variations of this technique have been alluded to by Grey [16], Covitz and Ashcraft [4], Sasaki and Smith [28], and Bowler [5]. We have already produced synthesized crescendos using a one-dimensional space of spectra, and we are now developing analysis software for multidimensional spaces.

6 CONCLUSIONS

We have described techniques for the automatic analysis and resynthesis of musical tones based on spectral interpolation. These techniques are interesting for several reasons. First, automatic analysis is important when it is desired to reproduce known sounds such as the sounds of traditional instruments. Second, we have achieved a high degree of data compression without perceptual degradation in quality or realism for a large and musically useful class of sounds. Third, the computation rate for synthesis is very low compared to other methods with equivalent generality. Finally, the data obtained from the analysis are in a form that can be modified and manipulated in various musically useful ways such as stretching, pitch changing, and interpolation between the spectra of different tones. We are currently studying an extension of this technique in which sequences of spectra are obtained not from a specific tone being reproduced, but by sampling an arbitrary trajectory in a precomputed spectral space. This extension promises a combination of simple and intuitive control, computational efficiency, and realistic production of traditional instrument tones.

7 ACKNOWLEDGMENT

The authors wish to acknowledge the countless people who shared their viewpoints and insights as this work progressed. They would in particular like to thank Richard Stern, Julius Smith, and Andy Moorer for their input. The Center for Art and Technology and the Computer Science Department both provided space and important organizational support, and Pamela Scott typed endless manuscript revisions.

8 REFERENCES

[1] J. A. Moorer, "Signal Processing Aspects of Computer Music—A Survey," *Comput. Music J.*, vol. 1, pp. 4–37 (1977 Feb.).

[2] J. M. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *J. Audio Eng. Soc.*, vol. 21, pp. 526–534 (1973 July/Aug.).

[3] H. Chamberlin, *Musical Applications of Microprocessors* (Hayden, Hasbrouck Heights, NJ, 1985).

[4] F. H. Covitz and A. C. Ashcraft, "Analysis and Generation of Complex Sounds Using Small Computers," in *Proc. Symp. on Small Computers in the Arts* (IEEE Computer Soc., 1981 Nov.).

[5] I. Bowler, "The Synthesis of Complex Audio Spectra by Cheating Quite a Lot," in *Proc. 1985 Int. Computer Music Conf.* (Computer Music Assoc. 1985 Aug.).

[6] G. Schwartz, "Sound Synthesis by Hierarchic Sampling," in *Proc. 1985 Int. Computer Music Conf.* (Computer Music Assoc. 1985 Aug.), pp. 62–73.

[7] J. L. Flanagan, in *Speech Analysis, Synthesis and Perception* (Springer, New York, 1972), chap. 8.

[8] D. Luce and M. Clark, "Physical Correlates of Brass-Instrument Tones," *J. Acoust. Soc. Am.*, vol. 42, no. 6 (1967).

[9] J. A. Moorer, "The Use of the Phase Vocoder in Computer Music Applications," *J. Audio Eng. Soc. (Engineering Reports)*, vol. 26, pp. 42–45 (1978 Jan./Feb.).

[10] M. Nikaido et al., "Wave Generating Method Apparatus Using Same," Matsushita Elect. Inc., U.S. Patent 4,597,318, filed 1984 Jan., issued 1986 July.

[11] R. E. Crochiere, "Optimum FIR Digital Filter Implementation for Decimation, Interpolation, and Narrow Band Filtering," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-23 (1975 Oct.).

[12] T. A. Ramstad, "Digital Methods for Conversion between Arbitrary Sampling Frequencies," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32 (1984 June).

[13] J. O. Smith, "A Flexible Sample Rate Conversion Method," in *Proc. 1984 ICASSP*, pp. 19.4.1–19.4.4.

[14] J. Moorer, "The Optimum Comb Method of Pitch Period Analysis of Continuous Digitized Speech," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-25 (1974 Oct.).

[15] A. V. Oppenheim, Ed., *Signal Processing: Digital Processing of Speech Signals* (Prentice-Hall, Englewood Cliffs, NJ, 1978).

[16] J. Grey, "An Exploration of Musical Timbre," Ph.D. thesis, Stanford University, Stanford, CA, 1975.

[17] J. Strawn, "Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis," *Comput. Music J.*, vol. 4, pp. 3–23 (1980 Fall).

[18] D. Rubine, "On the Analysis of Nearly Harmonic Tones," unpublished (1985).

[19] P. J. Comerford, "Bradford Musical Instrument Simulator," *Proc. IEE*, vol. 128 (1981 July).

[20] P. J. Comerford, "The Bradford Musical Instrument Simulator," in *Proc. 1986 Int. Computer Music Conf.* (Computer Music Assoc., 1986 Oct.).

[21] T. F. Quatieri and R. J. McAulnay, "Speech Transformations Based on Sinusoidal Representation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34 (1986).

[22] C. Roads, "Prophet VS: User's Report," *Comput. Music J.*, vol. 10, no. 2 (1987).

[23] J. Aikin, "Prophet VS Analog/Digital Synthesizer," *Keyboard* (1986 Aug.).

[24] C. Meyer, "Keytex CTS-2000 Crosstable Sampled Synthesizer," *Music Technol.* (1987).

[25] Y.-O. Lo, "Techniques of Timbral Interpolation," in *Proc. 1986 Int. Computer Music Conf.* (Computer Music Assoc., 1986).

[26] J. O. Smith and X. Serra, "PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds Based on a Sinusoidal Representation," in *Proc. 1987 Int. Computer Music Conf.* (Computer Music Assoc., 1987 Aug.).

[27] T. Greenwald, "Roland D-50 Digital Synthesizer," *Keyboard* (1987 Sept.).

[28] L. H. Sasaki and K. C. Smith, "A Simple Data Reduction Scheme for Additive Synthesis," *Comput. Music J.*, vol. 4, no. 1 (1980).

THE AUTHORS

M. H. Serra

D. Rubine

R. B. Dannenberg

Marie-Helene Serra received a diploma of engineering in physics in 1981 June from INSA (National Institute for Applied Sciences) in Toulouse, France, and a DEA (Diplome d'Etudes Approfondies) in acoustics from Bordeaux University in 1982 June. She received her doctoral thesis on speech synthesis in 1985 October and a diploma of docteur ingenieur from the Computer Science Department at Toulouse University.

Dr. Serra spent from 1985 November to 1987 November, at the Carnegie Mellon University in Pittsburgh, Pennsylvania, where she worked with Roger Dannenberg and Dean Rubine on sound synthesis via waveform interpolation. She is currently working at the CEMAMu center in Paris, which is directed by Iannis Xenakis, as a software engineer.

•

Dean Rubine received his B.S. and M.S. degrees in electrical engineering and computer science from MIT in 1982. He spent several years at Bell Laboratories, where he worked mainly on the implementation of programming languages. While at Bell, he implemented interpreters for Ada, Lisp, and Treat, an applicative language for expressing compilers. He also participated in the development of several Ada and C compilers.

Mr. Rubine has been a graduate student in the Computer Science Department of Carnegie Mellon University since 1983. Originally, he concentrated in the area of computer music, working on programming languages for real-time control, the analysis and synthesis of musical tones, and the creation of a new musical instrument. He now investigates novel forms of human-computer interaction, and has built computer systems able to interpret human gestures. He expects to complete his dissertation on gesture-based systems shortly.

•

Roger B. Dannenberg is a research computer scientist on the faculty of the Carnegie Mellon University School of Computer Science, where he received a Ph.D. in 1982. His current work includes research on computer accompaniment of live musicians, and the design and implementation of Arctic, a very high-level functional language for real-time control.

Dr. Dannenberg is also working in the CMU Studio for Creative Inquiry on the Piano Tutor, a multimedia expert system for teaching beginners to play the piano.