The Analysis and Resynthesis of Tones via Spectral Interpolation

Marie-Hélène Serra* Dean Rubine Roger B. Dannenberg

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This paper presents a new approach to the real-time generation of digital sounds. Our approach is based on the interpolation of spectra over time, and leads to a completely automated analysis/synthesis algorithm for the reproduction of natural sounds. The technique enables accurate reproduction of the spectral variations of acoustic instruments. Furthermore, spectral interpolation synthesis has a more efficient implementation than that of classical additive synthesis.

We have applied the spectral interpolation analysis/synthesis technique to different instruments and have obtained promising results. The technique greatly reduces the amount of data needed to represent a harmonic sound; however, it fails to convincingly reproduce inharmonic sounds. To reproduce sounds with inharmonic attacks we use a hybrid method which combines sampling and spectral interpolation synthesis.

1. Waveform Interpolation Synthesis

Waveform interpolation synthesis is a method for efficiently reproducing the spectral variation of acoustic instruments. Among existing real-time synthesis techniques we can place it between additive synthesis[14] and fixed-waveform synthesis[16,2]. Indeed, waveform interpolation synthesis can be viewed as a technique that reduces the cost of additive synthesis while allowing better quality and control than fixed-waveform synthesis.

*Current address: CeMaMu - CNET E655, 3 Avenue de la République, Issy les Moulineaux 92131, France

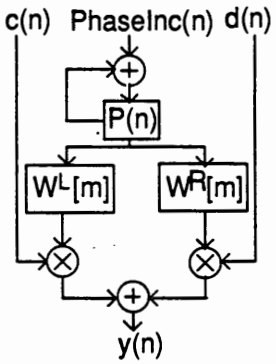

Figure 1: The Waveform Interpolation Oscillator

1.1. The Waveform Interpolation Oscillator

Waveform interpolation is similar in implementation to fixed-waveform synthesis but uses two phase-locked wavetables, W^L and W^R, each one loaded with a different waveform (Figure 1). The two tables have the same length M, and are indexed with the same phase value $P(n)$. Each wavetable has its own amplitude scaling factor. The results of the amplitude scaling are added together. The interpolation signal can be expressed as

$$y(n) = c(n)W^L[P(n)] + d(n)W^R[P(n)] \qquad (1)$$

where

n is the number (index) of the sample being computed,

Serra, Rubine, and Dannenberg, "The Analysis and Resynthesis of Tones via Spectral Interpolation," in Proceedings of the 1988 International Computer Music Conference, Cologne, West Germany, September 20-25, 1988. Eds. Christoph Lischka and Johannes Fritsch. San Francisco: International Computer Music Association, 1988. pp. 322-332.

$c(n)$ is the amplitude scale factor of the left wavetable at sample n,
$d(n)$ is that of the right,
$P(n)$ is the phase accumulator at sample n, and
$y(n)$ is the output signal at sample n.

A waveform interpolation oscillator is identical in effect to two table-lookup oscillators whose phases $P(n)$ are constrained to be equal.

A simple special case of equation (1) 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 write this special case of equation (1) as

$$y(n) = (1 - r(n))W^L[P(n)] + r(n)W^R[P(n)] \qquad (2)$$
$$r(n) = n/N$$

where N is the duration (in samples) of the synthesized signal $y(n)$. Intuitively we expect that the time-linear interpolation of the two waveforms is equivalent to the time-linear interpolation of their respective short-time spectra. Thus, we propose to generate spectral variation through waveform interpolation.

1.2. Spectral Evolution via Waveform Interpolation

We are going to use waveform interpolation (equation (2)) 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. To this end, we modify the wavetable interpolation oscillator so that it has pointers to the left and right wavetables rather than the wavetables themselves. We can then achieve complex spectral evolution by switching between wavetables over the course of a sound. By repeating the interpolation procedure (equation (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 ramp associated with the wavetable being changed is zero. As the ramp $r(n)$ and its opposite $(1 - r(n))$ are continuous the output of the oscillator will be free from clicks.

Figure 2 illustrates the time-linear 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 $(W_0, W_1), (W_1, W_2), \ldots, (W_{Q-2}, W_{Q-1})$.

As we see in the figure the scaling ramps $(1 - r(n))$ and $r(n)$ ($c(n)$ and $d(n)$ respectively) are piecewise-linear functions, each of which alternately has value zero at some n_i, rising linearly to unity at n_{i+1} and again reaching zero at n_{i+2}. To describe the waveform switching mathematically, we define two functions $L(n)$ and $R(n)$. Given the 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 changes of waveform for each table.

$$y(n) = (1 - r(n))W_{L(n)}[P(n)] + r(n)W_{R(n)}[P(n)] \qquad (3)$$
$$r(n) = \begin{cases} (n - n_{2i})/(n_{2i+1} - n_{2i}) & n_{2i} \le n < n_{2i+1} \\ (n_{2i} - n)/(n_{2i} - n_{2i-1}) & n_{2i-1} \le n < n_{2i} \end{cases}$$

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:

$$L(n) = 2i \quad \text{where} \quad n_{2i} \le n < n_{2i+2} \qquad (4)$$
$$R(n) = 2i + 1 \quad \text{where} \quad n_{2i-1} \le n < n_{2i+1}$$

The criteria to avoid clicks when switching waveforms are

$$r(n) = 0 \quad \text{when} \quad n = n_{2i} \qquad (5)$$
$$r(n) = 1 \quad \text{when} \quad n = n_{2i+1}$$

1.3. Spectral Interpolation Synthesis

Interpolating between waveforms is not necessarily equivalent to interpolating between their corresponding amplitude spectra. Since the spectra are complex, phase cancellation and phase shifting may result from interpolation. When corresponding harmonics in the spectra being interpolated are out of phase, the amplitudes of those harmonics do not change linearly with $r(n)$. Furthermore, such phase variations over time may result in an unintentional but perceptible timbre change, often perceived as a frequency shift. In order to get intuitive control over the interpolation process, we avoid these effects by interpolating between spectra whose corresponding harmonics

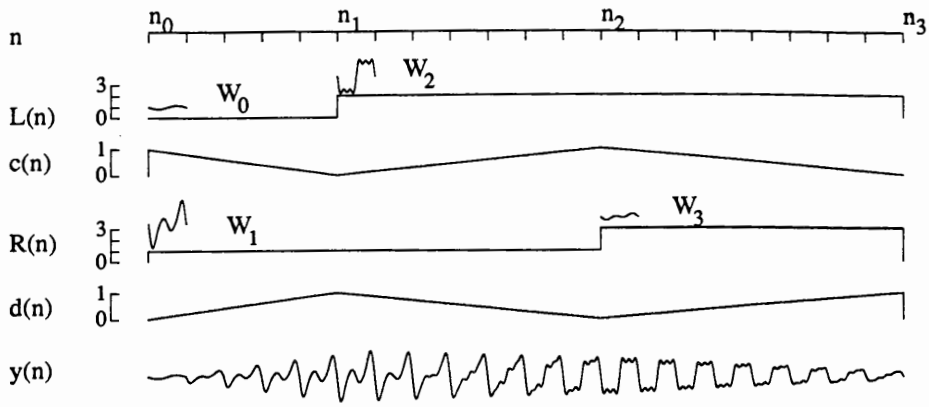

Figure 2: Interpolation of successive waveforms

are all in phase.[1] 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.

In spectral interpolation synthesis, we can express simply the result of the interpolation of two wavetables on the individual harmonics. Expressing the left and right wavetables of an interpolating oscillator as the sum of their respective harmonics, we have

$$W^L[m] = \sum_{h=0}^{H-1} a_h^L \cos(2\pi hm/H + \theta_h^L) \qquad (6)$$

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

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 (6) in (2) and assuming that the corresponding harmonics in W^L and W^R are in phase (*i.e.* $\theta_h^L = \theta_h^R = \theta_h$) we have

$$y(n) = \sum_{h=0}^{H-1} ((1 - r(n))a_h^L + r(n)a_h^R) \cos(2\pi hm/H + \theta_h) \quad (7)$$

Thus, the amplitude of the h_{th} harmonic in the output signal at sample n is

$$a_h(n) = (1 - r(n))a_h^L + r(n)a_h^R \qquad (8)$$

Thus, given the constraints of spectral interpolation, the amplitude of a harmonic of the output of the oscillator at sample n ramps linearly from its value in the left wavetable to its value in the right wavetable.

1.4. Reproducing a Tone via Spectral Waveform Interpolation

It is well known that a table lookup oscillator controlled with slowly varying functions can generate only harmonic sounds. The same result applies to the waveform interpolation oscillator.[2] Thus, we will be able to reproduce only harmonic sounds. To reproduce a sound with the generation model described by the equation (3), we perform a sequence of interpolations between pairs of waveforms. The waveforms $\{W_i\}$ used in equation (3) 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 number of

[1] As is often done in additive synthesis and the phase vocoder [15], we ignore the initial phases. We do, however, consider initial phase differences in Section 2.6..

[2] It is in fact possible to generate sounds with inharmonic partials by interpolation between waveforms containing out-of-phase harmonics [2], and this effect may be used, for example, to generate vibrato [17]. However we do not include this effect in the present study.

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*. The signal over a given interpolation interval is computed with the interpolation formula (3). To select the generator wavetables and the interpolation intervals, we need an analysis preceding the synthesis. The function of the analysis is to generate the control parameters of the spectral interpolation oscillator, such that the resynthesized signal sounds like the original.

2. 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. The analysis is performed in the spectral domain and is based on the interpolation equation between harmonics (equation (8)).

The analysis follows 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.

2.1. 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 separately study the reproduction of the amplitude spectral variations by spectral interpolation. Some modifications to the analysis and synthesis algorithms described here are required for them to work for tones whose pitch is not constant.

The sound coming from the instrument is digitally recorded using an A/D converter with a programmable sample rate and anti-aliasing filter. The sample rate is chosen so that there will be an integer number of samples per period[3] (see Section 2.2.).

[3]If it is not practical to record at arbitrary sample rates, an interpolation and decimation procedure [6] or other resampling algorithm [19,23] can change the sample rate efficiently.

2.2. Spectral Analysis

To measure the short-time spectrum, we use a pitch-synchronous Discrete Fourier Transform (DFT). In particular, we compute a DFT on each period of the tone. If the period is measured accurately, the DFT directly produces the amplitude and phase of each harmonic.[4] 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. As the signal is real, we have at most $\lfloor (P-1)/2 \rfloor$ harmonics (ignoring the DC component at $h = 0$). In actuality, we calculate H harmonics, $H \leq \lfloor (P-1)/2 \rfloor$, possibly ignoring some higher harmonics.[5]

Extracting the vector of amplitudes of the DFT on each period of the tone yields a list of spectra, one for each period. We call $S^{\langle i \rangle}$ the spectrum measured at period i. The total number of periods in the tone is denoted N_P.

$$S^{\langle i \rangle} = (a_1\langle i \rangle, a_2\langle i \rangle, \ldots, a_h\langle i \rangle, \ldots, a_H\langle i \rangle) \qquad 0 \leq i < N_P \tag{9}$$

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.

2.3. Spectral Ramps

In what follows we describe two 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 (equations (9), (6), (3)): a list of Q spectra with their associated times. 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.

[4]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 [13] and/or a simple peak detector.

[5]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 pitchs.

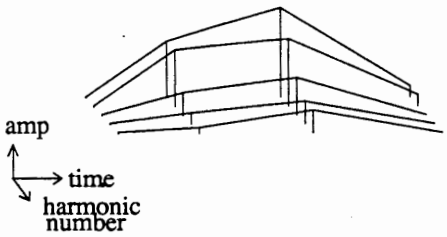

Figure 3: Three Spectral Ramps

In order to explain how the data reduction works, we first express the interpolation between two spectra $S^{(i)}$ and $S^{(j)}$ (successive in the synthesis) in terms of their individual harmonics.[6] Analogously to equation (8), the instantaneous interpolated harmonics at sample n within the spectral ramp $S^{(ij)}$, $a_h^{(ij)}(n)$, are expressed as[7]

$$a_h^{(ij)}(n) = (1 - r(n))a_h^{(i)} + r(n)a_h^{(j)} \qquad (10)$$
$$r(n) = \frac{n - n_i}{n_j - n_i}$$
$$n_i \leq n < n_j \quad 1 \leq h \leq H$$

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

$$S^{(ij)}(n) = (1 - r(n))S^{(i)} + r(n)S^{(j)} \qquad (11)$$
$$n_i \leq n < n_j$$

As 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, the sequence of spectra $S^{(ij)}(n), n_i \leq n < n_j$, is called the *spectral ramp* from $S^{(i)}$ to $S^{(j)}$. A spectral ramp consists of the H amplitude ramps which connect each harmonic of the initial spectrum $S^{(i)}$ to the harmonic of same order in the final spectrum $S^{(j)}$. 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 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 harmonics amplitudes. However, in contrast to the usual representations used for additive synthesis [8], 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 next section describes two different methods that are used to minimize our error criteria.

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

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. There are two different algorithms that compute the spectral ramps. The first algorithm 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 algorithm 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.

2.4.1. Spectral Ramp Interpolation Using Original Spectra

Consider the equations (10) and (11) which define the instantaneous 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 compare the successive interpolated 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 $n_j = jP$). The interpolated spectra are computed at the same rate as the DFT spectra, *i.e.* with a time interval equal to the period P, i.e $S^{(ij)}(l) = S^{(ij)}(n_l)$ with $n_l = lP$. Given this notation, the analysis equation

[6] 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 2.4.2.) computes the spectra used in the synthesis.

[7] For the remainder of Section 2. 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 $r(n)$ and $1 - r(n)$ must be interchanged.

(10) is rewritten as:

$$a_h^{(ij)}\langle l\rangle = a_h^{(i)} + \frac{n_l - n_i}{n_j - n_i}(a_h^{(j)} - a_h^{(i)}) \qquad i \le l < j \quad (12)$$

Comparing the amplitudes of the harmonics of spectrum $S^{(l)}$ with values given by equation (12) produces an error $E_l^{(ij)}$. $E_l^{(ij)}$ is defined as the sum of the squared errors on each of the H harmonics amplitudes:

$$E_l^{(ij)} = \sum_{h=1}^{H} (a_h^{(ij)}\langle l\rangle - a_h^{(l)})^2 \quad (13)$$

Using the following notation:

$$\begin{array}{ll} \Delta a_h^{(ij)} = a_h^{(j)} - a_h^{(i)} & \Delta a_h^{(il)} = a_h^{(l)} - a_h^{(i)} \\ \Delta n^{ij} = n_j - n_i & \Delta n^{il} = n_l - n_i \end{array} \quad (14)$$

combining equations (12) and (13) gives:

$$E_l^{(ij)} = \sum_{h=1}^{H} (\frac{\Delta n^{il}}{\Delta n^{ij}} \Delta a_h^{(ij)} - \Delta a_h^{(il)})^2 \quad (15)$$

The global error $E^{(ij)}$ within the spectral ramp $S^{(ij)}$ is defined as the sum of the errors on the individual spectra:[8]

$$E^{(ij)} = \sum_{l=i+1}^{j-1} E_l^{(ij)} \quad (16)$$

If the error $E^{(ij)}$ is less than the tolerated threshold E_{max} we extend the spectral ramp to the next period $\langle j+1\rangle$ and compute the new error $E^{(i,j+1)}$ using equation (16), $E_l^{(i,j+1)}$ being computed with equation (15). Otherwise we store the data defining the previous the spectral ramp $S^{(i,j-1)}$, and we compute the next ramp starting at spectrum $(n_{j-1}, S^{(j-1)})$.

2.4.2. Spectral Ramp Interpolation Using Computed Spectra

The *Linear Regression* algorithm is a way of fitting piecewise linear functions to a set of points [25]. We use a variant of linear regression called *anchored regression* [21]. 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 in general 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 (*i.e.* 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 $(n_i, a'_h\langle i\rangle)$ 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 $(n_{j+1}, S^{(j+1)})$ 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 $S'^{(j-1)}$, $a'_h\langle j-1\rangle$, $1 \le h \le H$, 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 $(n_i, a'_h\langle i\rangle)$ each line is defined by an equation of the form

$$a'^{(ij)}_h\langle l\rangle = m_h^{(ij)}(n_l - n_i) + a'_h\langle i\rangle \quad (17)$$
$$1 \le h \le H \quad i \le l \le j$$

where $m_h^{(ij)}$ is the slope of the h_{th} line of the ramp.

Defining $E_h^{(ij)}$ to be the sum of the squared errors of each harmonic within the ramp $S'^{(ij)}$, we have

$$E_h^{(ij)} = \sum_{l=i+1}^{j} (a_h\langle l\rangle - (m_h^{(ij)}(n_l - n_i) + a'_h\langle i\rangle))^2 \quad (18)$$

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

$$E^{(ij)} = \sum_{h=1}^{H} E_h^{(ij)} \quad (19)$$

[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 $E_i^{(ij)} = 0$.

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

By using the notation previously defined (equation (14)) with $a_h\langle i\rangle$ replaced by $a'_h\langle i\rangle$ we can write:

$$E^{(ij)} = \sum_{h=1}^{H} \sum_{l=i+1}^{j} (\Delta a_h^{(il)} - m_h^{(ij)} \Delta n^{il})^2 \qquad (20)$$

Setting $\frac{\partial E^{(ij)}}{\partial m_h^{(ij)}} = 0$ gives the slope of each line:

$$m_h^{(ij)} = \frac{\sum_{l=i+1}^{j} \Delta n^{il} \Delta a_h^{(il)}}{\sum_{l=i+1}^{j} (\Delta n^{il})^2} \qquad (21)$$

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 (20) and (21) when we add the new spectrum. We have an incremental algorithm (not shown for brevity) which 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 $E^{(ij)}$.

2.5. Results

After computing the spectral ramps (using either method), we resynthesize the tone by evaluating equations (6) and (2) in software. Each of the successive spectrum defining the spectral ramps is used to compute one period of the waveform defined in equation (6). The phases of the cosine waves θ_h are alternately set to $-\pi/2$ or $\pi/2$, so that the waveform and its first derivative are close to zero at the beginning and at the end of the table.[10]

After two waveforms defining a spectral ramp have been loaded in the wavetables, the wavetables are read, scaled by two opposite linear ramps (2) and added to form the output signal. The process is repeated until every spectral ramp has been synthesized.

The resemblance between the original signal and the resynthesized signal depends on the number of spectral ramps that are used to approximate the spectral envelope. The number of spectral ramps 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 two algorithms, (using original spectra and 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.

Figure 4 shows a sampled trombone tone.[11] 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.

From our small sample, we conclude that we can produce high quality harmonics sounds with a relatively small control bandwidth. The average number of waveforms per second in the synthesized tones is between 4 and 10. The average number of 8-bit bytes per second sent to the synthesizer is between 150 and 400. The synthesis takes approximately 10 operations per sample. We have compared the number of operations needed for additive synthesis and waveform interpolation synthesis (including, in the latter case, calculation of the wavetables) and we have come to the conclusion that waveform interpolation synthesis is most profitable when the number of harmonics being synthesized is large. Interestingly, waveform interpolation becomes relatively more efficient as the sampling rate increases. On the other hand, as the number of waveforms per second needed in the synthesis increases, and as the wavetable size increases, the relative efficiency of waveform interpolation decreases. Empirically we have found that if a given tone is able to be reproduced accurately by spectral interpolation, we can always do so at a cost less than that of additive synthesis.

As our analysis/synthesis method does not handle inharmonicity in general, the attacks of some tones (es-

[10] This is the technique used in the Bradford Musical Instrument Simulator [3].

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

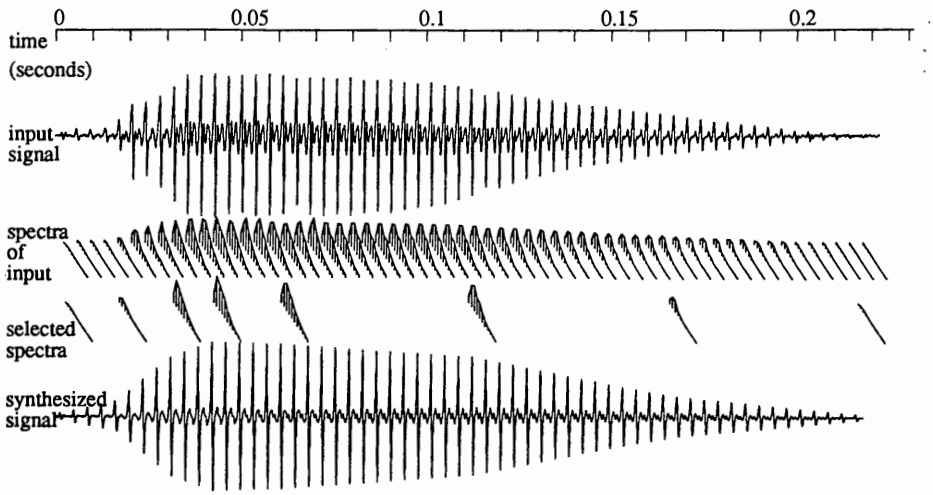

Figure 4: Synthesis by Time Linear Spectral Interpolation

pecially brass instruments) were not synthesized with enough realism.[12] 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 and evaluate in the next section.

2.6. Combining 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 gotten by analyzing the tone using one of our algorithms previously described.[13] We discuss the techniques we tried for connecting the sampled and synthesized portions.

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.[14] 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 $\pm\pi/2$.[15] We wanted to find a way of shifting the initial phases of the harmonics so that the frequency shift resulting from the phase shift is imperceptible. We tried to use a phase interpolation algorithm, based on [18], where the phases are progressively interpolated (using cubic polynomials) from their original value to $\pm\pi/2$. Unfortunately, the phase interpolation was always perceived as a frequency shift. While the small frequency shift was unnoticeable when synthesizing speech[18], when reproducing isolated tones such a shift becomes suprisingly obvious.

[12] Both the pitch-synchronous DFT and the interpolation synthesis cannot deal with inharmonicity.

[13] 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.

[14] The alternative of crossfading between sampled and synthesized signals without regard to phase has been explored and rejected by Smith and Serra[10]. Our own experiments confirm the difficulty. Interestingly, the Ensoniq SQ80[24] sampling synthesizer claims to employ cross-fading successfully.

[15] We had an ulterior motive for trying to do this. We wish to use the Bradford Musical Instrument Simulator [4] hardware for our synthesis. This hardware currently requires the phases of the harmonics in the wavetable to be $\pm\pi/2$. A number of benefits come as a consequence of this restriction: the transition between successive wavetable 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.

Thus, we reverted to a simple phase matching technique. We use the phases extracted from the last period of the attack to compute every waveform used during the synthesis (equation (6)). By doing this, the transition between sampled and synthesized signal is faultless. 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 and woodwind instruments. Figure 5 shows the same input tone as the one previous figure synthesized using a sampled attack. The arrow in the figure points to the transition between sampled sound and synthesized sound.

In practice we find that we need from 30 to 60 milliseconds of the sampled attack for good results (500 to 1000 samples at a 16 KHz sample rate). The typical data rate needed to control the synthesizer, excluding transmission of the sampled attack, ranges from 40 to 400 bytes per second. 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.

3. Related Work and Future Plans

We have presented and discussed the technique of spectral interpolation for the analysis and resynthesis acoustic instruments. We use a succession of wavetables that are dynamically mixed to reproduce analyzed spectral evolutions.

The analysis algorithm we presented in Section 2.4. used linear interpolation in time to gradually change from one waveform to the next. We have another algorithm called *nonlinear interpolation* in which the scaling factors $c(n)$ and $d(n)$ (equation (1)) of the two wavetables are arbitrary. Nonlinear interpolation generally reduces the minimum number of spectra needed to represent a given tone.

We have simplified our analysis by restricting our data to relatively fixed-frequency examples. To relax this restriction, it should be possible to resample the input to obtain a fixed number of samples per period, perform the analysis, and then use a time-varying frequency in the table lookup oscillator to reproduce the original frequency.[16]

The idea of mixing multiple waveforms to generate sounds whose spectra change over time is not new. Several commercial synthesizers have implemented this technique, among which the Matsushita digital instrument [17], the Prophet VS [20,1] and the Keytex CTS-2000 [12]. The contribution of our work is that we provide an automatic analysis to make possible the resynthesis of acoustic instruments.

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 multi-dimensional space of spectra. Typical dimensions would be amplitude and frequency, which are input parameters to this synthesis model. Other parameters, such as bow position, lip pressure, etc. 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 [8], Covitz and Ashcraft [5], Sazaki and Smith [22], Bowler [9], Comerford [3,4], and Lo [11].

J. Smith and X. Serra [10] independently combined sampled attacks with synthesized tones using the same phase-matching technique described in Section 2.6.. Their synthesis technique was based on summation of sinusoids rather than interpolation. The Roland D-50 [7] 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 insure smooth transitions.

4. Conclusions

We have described a technique for the automatic analysis and resynthesis of musical tones based on spectral interpolation. This technique is 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 the synthesis is low compared to other methods with equivalent

[16] To resample, the time-varying frequency must first be determined by some pitch extraction method. The signal can then be resampled [23] at a sampling rate that varies in proportion to the measured frequency.

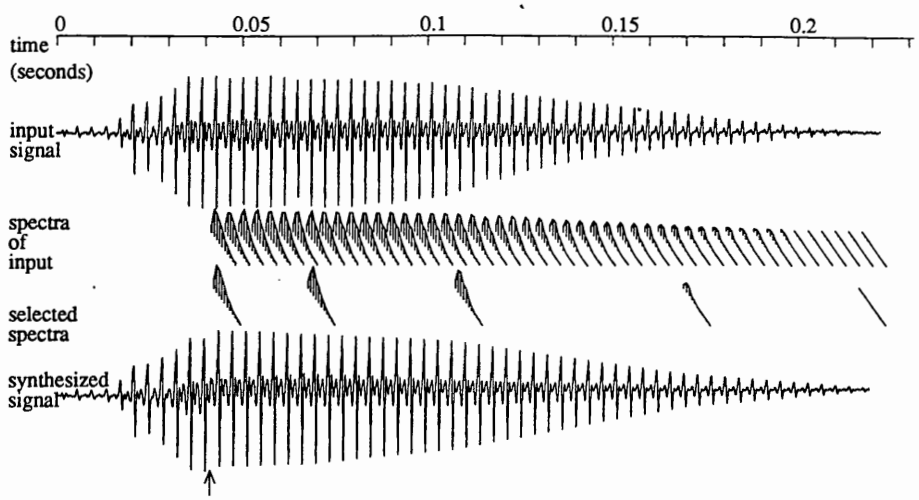

Figure 5: Spectral Interpolation with Sampled Attack

generality. Finally, the data obtained from the analysis is 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.

5. Acknowledgments

We gratefully acknowledge the support of Apple Computer, Inc., and the Yamaha Corporation, whose grants enabled the undertaking of this work. We would also like to credit the Defense Advanced Research Projects Agency (DOD) for additional support, 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 an IBM Fellowship Supplement and a Ben Franklin Partnership Fund Matching Award, Agreement Number 402995-45176-308-001. The Center for Art and Technology and the Computer Science Department both provided space and important organizational support.

Countless people have shared their viewpoints and insights as this work progressed. We would particularly like to thank Richard Stern, Julius Smith, and Andy Moorer for their input.

References

[1] J. Aikin. Prophet VS analog/digital synthesizer. Keyboard magazine, August 1986.

[2] Hal Chamberlin. *Musical Applications of Microprocessors.* Hayden Books, 1985.

[3] P.J. Comerford. Bradford musical instrument simulator. *IEE Proc.*, 128(5), July 1981.

[4] P.J. Comerford. The Bradford musical instrument simulator. In *Proceedings of the 1986 International Computer Music Conference*, October 1986.

[5] F.H. Covitz and A.C. Ashcraft. Analysis and generation of complex sounds using small computers. In *Proceedings of Symposium on Small Computers in the Arts*, IEEE Computer Society, November 1981.

[6] R.E. Crochiere. Optimum FIR digital filter implementation for decimation, interpolation, and narrow band filtering. *IEEE Transactions on ASSP*, ASSP-23(5), October 1975.

[7] T. Greenwald. Roland D-50 digital synthesizer. Keyboard magazine, September 1987.

[8] John M. Grey. *An Exploration of Musical Timbre*. PhD thesis, Stanford University, 1975.

[9] I.Bowler. The synthesis of complex audio spectra by cheating quite a lot. In *Proceedings of the 1985 International Computer Music Conference*, Computer Music Association, August 1985.

[10] J.O.Smith and X.Serra. PARSHL: an analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation. In *Proceedings of the 1987 International Computer Music Conference*, Computer Music Association, August 1987.

[11] Yee-On Lo. Techniques of timbral interpolation. In *Proceedings of the 1986 International Computer Music Conference*, Computer Music Association, 1986.

[12] C. Meyer. Keytex CTS-2000 crosstable sampled synthesizer. Music Technology magazine, 1987.

[13] J. Moorer. The optimum comb method of pitch period analysis of continuous digitized speech. *IEEE Trans. on ASSP*, ASSP-22(5), October 1974.

[14] James A. Moorer. Signal processing aspects of computer music - a survey. *Computer Music Journal*, 1(1):4–37, February 1977.

[15] James A. Moorer. The use of the phase vocoder in computer music applications. *Journal of the Audio Engineering Society*, 26(1/2):42–45, February 1978.

[16] M.V. Mathews with J.E.Miller, F.R.Moore, J.C.Risset. and J.R.Pierce *The Technology of Computer Music*. MIT Press, 1969.

[17] M. Nikaido. Wave generating method apparatus using same. Matsushita Elect. Inc., Filed in January 1984 Issued in July 1986.

[18] T.F. Quatieri and R.J. McAulnay. Speech transformations based on sinusoidal representation. *IEEE Trans on ASSSP*, ASSP-34, 1986.

[19] T.A. Ramstad. Digital methods for conversion between arbitrary sampling frequencies. *IEEE Transactions on ASSP*, ASSP-32(3), June 1984.

[20] Curtis Roads. Prophet VS: user's report. Computer Music Journal Vol.10, Number 2, 1987.

[21] Dean Rubine. On the analysis of nearly harmonic tones. Unpublished 1985.

[22] L.H. Sasaki and K.C. Smith. A simple data reduction scheme for additive synthesis. *Computer Music Journal*, 4(1), 1980.

[23] J.O. Smith. A flexible sample rate conversion method. In *Proceedings of ICASSP*, pages 19.4.1–19.4.4, 1984.

[24] Ensoniq SQ80. Cross wave synthesis. *Electronic Musician*, 4(1):69, January 1988.

[25] John Strawn. Approximation and syntactic analysis of amplitude and frequency functions for digital sound synthesis. *Computer Music Journal*, 4(3):3–23, Fall 1980.