PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds Based on a Sinusoidal Representation*

Julius O. Smith III†(jos@ccrma.stanford.edu)
Xavier Serra (xjs@ccrma.stanford.edu)
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

Abstract

This paper describes a peak-tracking spectrum analyzer, called PARSHL, which is useful for extracting additive synthesis parameters from inharmonic sounds such as the piano. PARSHL is based on the Short-Time Fourier Transform (STFT), adding features for tracking the amplitude, frequency, and phase trajectories of spectral lines from one FFT to the next. PARSHL can be thought of as an "inharmonic phase vocoder" which uses tracking vocoder analysis channels instead of a fixed harmonic filter bank as used in previous FFT-based vocoders.

*This is the original full version from which the Technical Report (CCRMA STAN-M-43) and conference paper (ICMC-87) were prepared. Additionally, minor corrections are included, and a few pointers to more recent work have been added.

†Work supported in part by Dynacord, Inc., 1985

Contents

1 Introduction and Overview

Short-Time Fourier Transform (STFT) techniques [1, 3, 2, 5, 17, 18, 19] are widely used in computer music applications [6, 13] for analysis-based additive synthesis. With these techniques, the signal is modeled as a sum of sine waves, and the parameters to be determined by analysis are the slowly time-varying amplitude and frequency for each sine wave.

In the following subsections, we will review the short-time Fourier transform, the phase vocoder, additive synthesis, and overlap-add synthesis. We then close the introduction with an outline of the remainder of the paper.

The Short-Time Fourier Transform (STFT)

Computation of the STFT consists of the following steps:

1. Read M samples of the input signal x into a local buffer,

$$x_m(n) \stackrel{\Delta}{=} x(n - mR), \qquad n = -M_h, -M_h + 1, \ldots, -1, 0, 1, \ldots, M_h - 1, M_h$$

where x_m is called the mth *frame* of the input signal, and $M \stackrel{\Delta}{=} 2M_h + 1$ is the frame length (which we assume is *odd* for reasons to be discussed later). The time advance R (in samples) from one frame to the next is called the *hop size.*

2. Multiply the data frame pointwise by a length M spectrum analysis window $w(n), n = -M_h, \ldots, M_h$ to obtain the mth windowed data frame:

$$\tilde{x}_m(n) \stackrel{\Delta}{=} x_m(n)w(n), \qquad n = -\frac{M-1}{2}, \ldots, \frac{M-1}{2}$$

3. Extend $\tilde{x}_m$ with zeros on both sides to obtain a *zero-padded* windowed data frame:

$$\tilde{x}'_m(n) \stackrel{\Delta}{=} \begin{cases} \tilde{x}_m(n), & |n| \leq \frac{M-1}{2} \\ 0, & \frac{M-1}{2} < n \leq \frac{N}{2} - 1 \\ 0, & -\frac{N}{2} \leq n < -\frac{M-1}{2} \end{cases}$$

where N is the FFT size, chosen to be a power of two larger than M. The number N/M is called the *zero-padding factor.*

4. Take a length N FFT of $\tilde{x}_m$ to obtain the STFT at time m:

$$\tilde{x}'_m(e^{j\omega_k}) = \sum_{n=-N/2}^{N/2-1} \tilde{x}'_m(n)e^{-j\omega_k nT}$$

where $\omega_k = 2\pi k f_s/N$, and $f_s = 1/T$ is the sampling rate in Hz. The STFT *bin number* is k. Each bin $\tilde{x}'_m(e^{j\omega_k})$ of the STFT can be regarded as a sample of the complex signal at the output of a lowpass filter whose input is $\tilde{x}'_m(n)e^{-j\omega_k mT}$; this signal is $\tilde{x}'_m(n)$ frequency-shifted so that frequency ω_k is moved to 0 Hz. In this interpretation, the hop size R is the *downsampling factor* applied to each bandpass output, and the analysis window $w(\cdot)$ is the *impulse response of the anti-aliasing filter* used with the downsampling.

The zero-padding factor is the *interpolation factor* for the spectrum, i.e., each FFT bin is replaced by N/M bins, interpolating the spectrum.

The Phase Vocoder

The steps normally taken by a "phase vocoder" to measure instantaneous amplitude and frequency for each bin of the current STFT frame are as follows (extending the four steps of the previous section):

5. Convert each FFT bin $\tilde{x}'_m(e^{j\omega_k})$ from rectangular to polar form to get the magnitude and phase in each FFT bin, and differentiate the unwrapped phase to obtain *instantaneous frequency:*

$$A_k(m) \triangleq \left|\tilde{x}'_m(e^{j\omega_k})\right| \tag{1}$$

$$\Theta_k(m) \triangleq \angle\tilde{x}'_m(e^{j\omega_k}) \qquad \text{(radians)} \tag{2}$$

$$F_k(m) \triangleq \frac{\Theta_k(m) - \Theta_k(m-1)}{2\pi RT} \qquad \text{(Hz)} \tag{3}$$

Additive Synthesis

To obtain oscillator-control envelopes for additive synthesis, the amplitude, frequency, and phase trajectories are estimated once per FFT hop by the STFT. It is customary in computer music to *linearly interpolate* the amplitude and frequency trajectories from one hop to the next. Call these signals $\hat{A}_k(n)$ and $\hat{F}_k(n)$, defined now for all n at the normal signal sampling rate. The phase is usually discarded at this stage and redefined as the integral of the instantaneous frequency when needed: $\hat{\Theta}_k(n) \triangleq \hat{\Theta}_k(n-1) + 2\pi T\hat{F}_k(n)$. When phase must be matched in a given frame, the frequency can instead move quadratically across the frame to provide cubic polynomial phase interpolation [12], or a second linear breakpoint can be introduced somewhere in the frame for the frequency trajectory.

6. Apply any desired modification to the analysis data, such as time scaling, pitch transposition, formant modification, etc.

7. Use the (possibly modified) amplitude and frequency trajectories to control a summing oscillator bank:

$$\hat{x}(n) \triangleq \frac{1}{N}\sum_{k=-N/2+1}^{N/2-1} \hat{A}_k(n)e^{j\hat{\Theta}_k(n)} \tag{4}$$

$$= \frac{2}{N}\sum_{k=0}^{N/2-1} \hat{A}_k(n)\cos(\hat{\Theta}_k(n)) \tag{5}$$

Overlap-Add Synthesis

A less computationally expensive alternative to sinusoidal summation is called *overlap-add* reconstruction [1, 3] which consists of the following steps:

6. Apply any desired modification to the spectra, such as multiplying by a filter frequency response function, to obtain the modified frame spectrum $\hat{X}'_m$. Additionally, desired spectral components can be added to the FFT buffer [4, 21].

7. Inverse FFT $\hat{X}'_m$ to obtain the windowed output frame:

$$\hat{x}'_m(n) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \hat{X}'_m(e^{j\omega_k}) e^{j\omega_k n}$$

8. Reconstruct the final output by overlapping and adding the windowed output frames:

$$\hat{x}(n) = \sum_m \hat{x}'_m(n - mR)$$

Analysis and resynthesis by overlap-add (in the absence of spectral modifications) is an *identity operation* if the overlapped and added analysis windows sum to unity, i.e., if

$$A_w(n) \stackrel{\Delta}{=} \sum_{m=-\infty}^{\infty} w(n - mR) = 1 \tag{6}$$

for every n. If the overlap-added window function $A_w(n)$ is not constant, it is then an *amplitude modulation* envelope with period R. That is, when the analysis window does not displace and add to a constant, the output is amplitude modulated by a periodic signal having its fundamental frequency at the frame rate f_s/R. Frame rate distortion of this kind may be seen as AM sidebands with spacing f_s/R in a spectrogram of the output signal. Not too surprisingly, condition Eq. (6) can be shown (by means of the "digital Poisson summation formula" [16]) to be equivalent to the condition that $W(e^{j\omega_k})$ be 0 at all harmonics of the frame rate f_s/R.

Parshl

Parshl performs data-reduction on the STFT appropriate for inharmonic, quasi-sinusoidal-sum signals. The goal is to track only the most prominent peaks in the spectrum of the input signal, sampling the amplitude approximately once per period of the lowest frequency in the analysis band. Parshl will do either additive synthesis or overlap-add synthesis, or both, depending on the application.

An outline of Parshl appears in §2, and Sections 3 to 8 discuss parameter-selection and algorithmic issues. Section 9 discusses analysis and resynthesis without phase information. Section 10 centers on the preprocessing of the input signal for better analysis/synthesis results. In §11 some applications are mentioned.

2 Outline of the Program

Parshl follows the amplitude, frequency, and phase[1] of the most prominent peaks over time in a series of spectra, computed using the Fast Fourier Transform (FFT). The synthesis part of the program uses the analysis parameters, or their modification, to generate a sinewave for every peak track found.

The steps carried out by Parshl are as follows:

[1]The version written in 1985 did not support phase. Phase support was added much later by the second author in the context of his Ph.D. research, based on the work of McAulay and Quatieri [12].

1. Compute the STFT $\tilde{x}'_m(e^{j\omega_k})$ using the frame size, window type, FFT size, and hop size specified by the user.

2. Compute the squared magnitude spectrum in dB ($20\log_{10}\left|\tilde{x}'_m(e^{j\omega_k})\right|$).

3. Find the bin numbers (frequency samples) of the spectral peaks. Parabolic interpolation is used to refine the peak location estimates. Three spectral samples (in dB) consisting of the local peak in the FFT and the samples on either side of it suffice to determine the parabola used.

4. The magnitude and phase of each peak is calculated from the maximum of the parabola determined in the previous step. The parabola is evaluated separately on the real and imaginary parts of the spectrum to provide a complex interpolated spectrum value.

5. Each peak is assigned to a frequency track by matching the peaks of the previous frame with the current one. These tracks can be "started up," "turned-off" or "turned-on" at any frame by ramping in amplitude from or toward 0.

6. Arbitrary modifications can be applied to the analysis parameters before resynthesis.

7. If additive synthesis is requested, a sinewave is generated for each frequency track, and all are summed into an output buffer. The instantaneous amplitude, frequency, and phase for each sinewave are calculated by interpolating the values from frame to frame. The length of the output buffer is equal to the hop size R which is typically some fraction of the window length M.

8. Repeat from step 1, advancing R samples each iteration until the end of the input sound is reached.

3 Analysis Window (Step 1)

The choice of the analysis window is important. It determines the trade-off of time versus frequency resolution which affects the smoothness of the spectrum and the detectability of the frequency peaks. The most commonly used windows are called Rectangular, Triangular, Hamming, Hanning, Kaiser, and Chebyshev. Harris [7, 14] gives a good discussion of these windows and many others.

To understand the effect of the window lets look at what happens to a sinusoid when we Fourier transform it. A complex sinusoid of the form

$$x(n) = Ae^{j\omega_x nT}$$

when windowed, transforms to

$$\begin{aligned} X_w(\omega) &= \sum_{n=-\infty}^{\infty} x(n)w(n)e^{-j\omega nT} && (7)\\ &= A\sum_{n=-(M-1)/2}^{(M-1)/2} w(n)e^{-j(\omega-\omega_x)nT} && (8)\\ &= AW(\omega-\omega_x) && (9) \end{aligned}$$

Thus, the transform of a windowed sinusoid, isolated or part of a complex tone, is the transform of the window scaled by the amplitude of the sinusoid and centered at the sinusoid's frequency.

All the standard windows are real and symmetric and have spectra of a sinc-like shape (as in Fig. 1). Considering the applications of the program, our choice will be mainly determined by two of the spectrum's characteristics: the width of the main lobe, defined as the number of bins (DFT-sample points) between the two zero crossings, and the highest side-lobe level, which

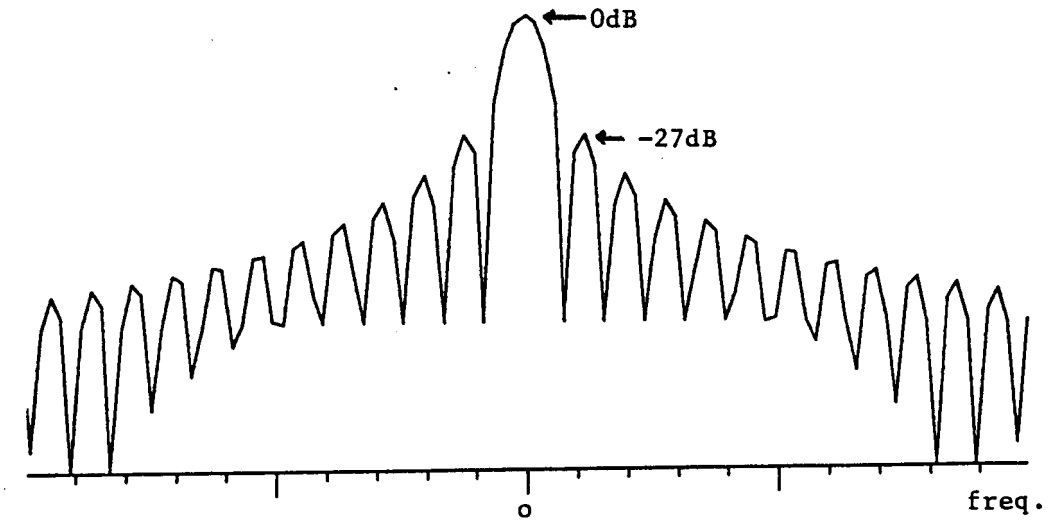

Figure 1: Log magnitude of the transform of a triangle window.

measures how many dB down is the highest side-lobe from the main lobe. Ideally we would like a narrow main lobe (good resolution) and a very low side-lobe level (no cross-talk between FFT channels). The choice of window determines this trade-off. For example, the rectangular window has the narrowest main lobe, 2 bins, but the first side-lobe is very high, -13dB relative to the main-lobe peak. The Hamming window has a wider main lobe, 4 bins, and the highest side-lobe is 42dB down. The Blackman window worst-case side-lobe rejection is 58 dB down which is good for audio applications. A very different window, the Kaiser, allows control of the trade-off between the main-lobe width and the highest side-lobe level. If we want less main-lobe width we will get higher side-lobe level and vice versa. Since control of this trade-off is valuable, the Kaiser window is a good general-purpose choice.

Let's look at this problem in a more practical situation. To "resolve" two sinusoids separated in frequency by Δ Hz, we need (in noisy conditions) two clearly discernible main lobes; i.e., they should look something like in Fig. 2. To obtain the separation shown (main lobes meet near a 0-crossing), we require a main-lobe bandwidth B_f in Hz such that

$$B_f \leq \Delta.$$

In more detail, we have

$$B_f = K\frac{f_s}{M} \tag{10}$$

$$\Delta = f_2 - f_1 \tag{11}$$

where K is the main-lobe bandwidth (in bins), f_s the sampling rate, M is the window length, and f_1, f_2 are the frequencies of the sinusoids. Thus, we need

$$M \geq K\frac{f_s}{\Delta} = K\frac{f_s}{f_2 - f_1}$$

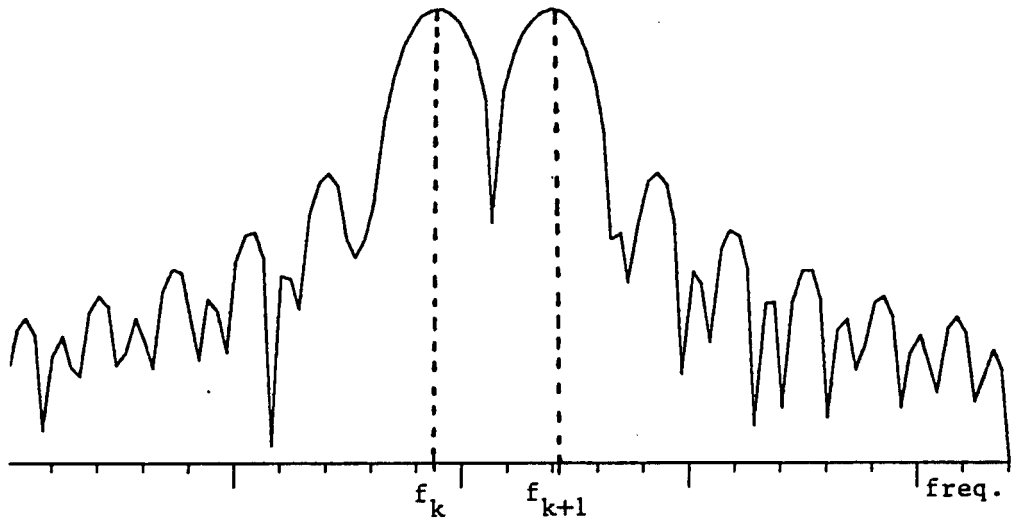

Figure 2: Spectrum of two clearly separated sinusoids.

If f_k and f_{k+1} are successive harmonics of a fundamental frequency f_1, then $f_1 = f_{k+1} - f_k = \Delta$. Thus, harmonic resolution requires $B_f \leq f_1$ and thus $M \geq K f_s/f_1$. Note that $f_s/f_1 = T_1/T = P$, the period in samples. Hence,

$$M \geq KP$$

Thus, with a Hamming window, with main-lobe bandwidth $K = 4$ bins, we want at least four periods of a harmonic signal under the window. More generally, for two sinusoids at any frequencies f_1 and f_2, we want four periods of the difference frequency $|f_2 - f_1|$ under the window.

While the main lobe should be narrow enough to resolve adjacent peaks, it should not be narrower than necessary in order to maximize time resolution in the STFT.

Since for most windows the main lobe is much wider than any side lobe, we can use this fact to avoid spurious peaks due to side-lobes oscillation. Any peak that is substantially narrower than the main-lobe width of the analysis window will be rejected as a local maximum due to side-lobe oscillations.

A final point we want to make about windows is the choice between odd and even length. An odd length window can be centered around the middle sample, while an even length one does not have a mid-point sample. If one end-point is deleted, an odd-length window can be overlapped and added so as to satisfy Eq. (6). For purposes of phase detection, we prefer a zero-phase window spectrum, and this is obtained most naturally by using a symmetric window with a sample at the time origin. We therefore use odd length windows exclusively in PARSHL.

Choice of Hop Size

Another question related to the analysis window is the hop size R, i.e., how much we can advance the analysis time origin from frame to frame. This depends very much on the purposes of the analysis. In general, more overlap will give more analysis points and therefore smoother results across time, but the computational expense is proportionately greater. For purposes of spectrogram display or additive synthesis parameter extraction, criterion Eq. (6) is a good general purpose choice. It states

that the succesive frames should overlap in time in such a way that all data are weighted equally. However, it can be overly conservative for steady-state signals. For additive synthesis purposes, it is more efficient and still effective to increase the hop size to the number of samples over which the spectrum is not changing appreciably. In the case of the steady-state portion of piano tones, the hop size appears to be limited by the fastest amplitude envelope "beat" frequency caused by mistuning strings on one key or by overlapping partials from different keys.

For certain window types (sum-of-cosine windows), there exist perfect overlap factors in the sense of Eq. (6). For example, a Rectangular window can hop by M/k, where k is any positive integer, and a Hanning or Hamming window can use any hop size of the form $(M/2)/k$. For the Kaiser window, on the other hand, there is no perfect hop size other than $R = 1$.

The perfect overlap-add criterion for windows and their hop sizes is not the best perspective to take when overlap-add synthesis is being constructed from the modified spectra $\tilde{x}'_m(e^{j\omega_k})$ [1]. As mentioned earlier, the hop size R is the downsampling factor applied to each FFT filter-bank output, and the window is the envelope of each filter's impulse response. The downsampling by R causes *aliasing,* and the frame rate f_s/R is equal to twice the "folding frequency" of this aliasing. Consequently, to minimize aliasing, the choice of hop size R should be such that the folding frequency exceeds the "cut-off freqency" of the window. The cut-off frequency of a window can be defined as the frequency above which the window transform magnitude is less than or equal to the worst-case sidelobe level. For convenience, we typically use the frequency of the first zero-crossing beyond the main lobe as the definition of cut-off frequency. Following this rule yields 50% overlap for the rectangular window, 75% overlap for Hamming and Hanning windows, and 83% (5/6) overlap for Blackman windows. The hop size useable with a Kaiser window is determined by its design parameters (principally, the desired time-bandwidth product of the window, or, the "beta" parameter) [8].

One may wonder what happens to the aliasing in the perfect-reconstruction case in which Eq. (6) is satisfied. The answer is that aliasing does occur in the individual filter-bank outputs, but this aliasing is canceled in the reconstruction by overlap-add *if* there were no modifications to the STFT. For a general discussion of aliasing cancellation in downsampled filter banks, see [23, 24].

4 Filling the FFT Input Buffer (Step 2)

The FFT size N is normally chosen to be the first power of two that is at least twice the window length M, with the difference $N - M$ filled with zeros ("zero-padded"). The reason for increasing the FFT size and filling in with zeros is that zero-padding in the time domain corresponds to interpolation in the frequency domain, and interpolating the spectrum is useful in various ways. First, the problem of finding spectral peaks which are not exact bin frequencies is made easier when the spectrum is more densely sampled. Second, plots of the magnitude of the more smoothly sampled spectrum are less likely to confuse the untrained eye. (Only signals truly periodic in M samples should not be zero-padded. They should also be windowed only by the Rectangular window.) Third, for overlap-add synthesis from spectral modifications, the zero-padding allows for multiplicative modification in the frequency domain (convolutional modification in the time domain) without time aliasing in the inverse FFT. The length of the allowed convolution in the time domain (the impulse response of the effective digital filter) equals the number of extra zeros (plus one) in the zero padding.

If K is the number of samples in the main lobe when the zero-padding factor is 1 ($N = M$),

then a zero-padding factor of N/M gives KN/M samples for the same main lobe (and same main-lobe bandwidth). The zero-padding (interpolation) factor N/M should be large enough to enable accurate estimation of the true maximum of the main lobe after it has been frequency shifted by some arbitrary amount equal to the frequency of a sinusoidal component in the input signal. We have determined by computational search that, for a rectangularly windowed sinusoid (of any frequency), quadratic frequency interpolation (using the three highest bins) yields at least 0.1% (of the distance from the sinc peak to the first zero-crossing) accuracy if the zero-padding factor N/M is 5 or higher.

As mentioned in the previous section, we facilitate phase detection by using a zero-phase window, i.e., the windowed data (using an odd length window) is centered about the time origin. A zero-centered, length M data frame appears in the length N FFT input buffer as shown in Fig. 3c. The first $(M-1)/2$ samples of the windowed data, the "negative-time" portion, will be stored at the end of the buffer, from sample $N-(M-1)/2$ to $N-1$, and the remaining $(M+1)/2$ samples, the zero- and "positive-time" portion, will be stored starting at the beginning of the buffer, from sample 0 to $(M-1)/2$. Thus, all zero padding occurs in the *middle* of the FFT input buffer.

5 Peak Detection (Steps 3 and 4)

Due to the sampled nature of spectra obtained using the STFT, each peak (location and height) found by finding the maximum-magnitude frequency bin is only accurate to within half a bin. A bin represents a frequency interval of f_s/N Hz, where N is the FFT size. Zero-padding increases the number of FFT bins per Hz and thus increases the accuracy of the simple peak detection. However, to obtain frequency accuracy on the level of 0.1% of the distance from a sinc maximum to its first zero crossing (in the case of a rectangular window), the zero-padding factor required is 1000. (Note that with no zero padding, the STFT analysis parameters are typically arranged so that the distance from the sinc peak to its first zero-crossing is equal to the fundamental frequency of a harmonic sound. Under these conditions, 0.1% of this interval is equal to the relative accuracy in the fundamental frequency measurement. Thus, this is a realistic specification in view of pitch discrimination accuracy.) Since we would nominally take two periods into the data frame (for a Rectangular window), a 100 Hz sinusoid at a sampling rate of 50 KHz would have a period of $50,000/100 = 500$ samples, so that the FFT size would have to exceed one million. A more efficient spectral interpolation scheme is to zero-pad only enough so that quadratic (or other simple) spectral interpolation, using only bins immediately surrounding the maximum-magnitude bin, suffices to refine the estimate to 0.1% accuracy. PARSHL uses a parabolic interpolator which fits a parabola through the highest three samples of a peak to estimate the true peak location and height (cf. Fig. 4).

We have seen that each sinusoid appears as a shifted window transform which is a sinc-like function. A robust method for estimating peak frequency with very high accuracy would be to fit a window transform to the sampled spectral peaks by cross-correlating the whole window transform with the entire spectrum and taking and interpolated peak location in the cross-correlation function as the frequency estimate. This method offers much greater immunity to noise and interference from other signal components.

To describe the parabolic interpolation strategy, let's define a coordinate system centered at $(k_\beta, 0)$, where k_β is the bin number of the spectral magnitude maximum, i.e., $\tilde{x}'_m(e^{j\omega_{k_\beta}}) \geq \tilde{x}'_m(e^{j\omega_k})$

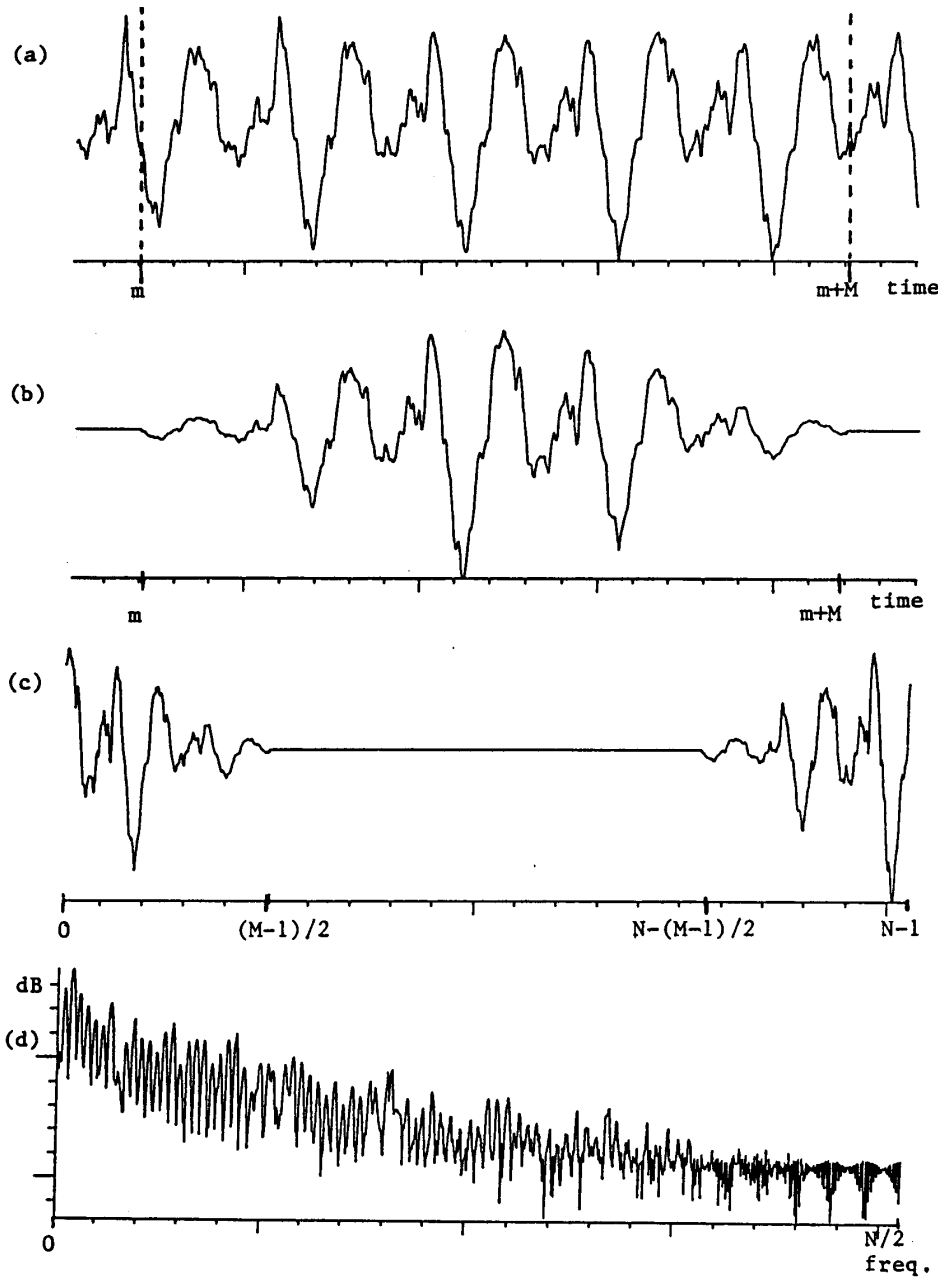

Figure 3: Illustration of the first two steps of PARSHL. (a) Input data. (b) Windowed input data. (c) FFT buffer with the windowed input data. (d) Resulting magnitude spectrum.

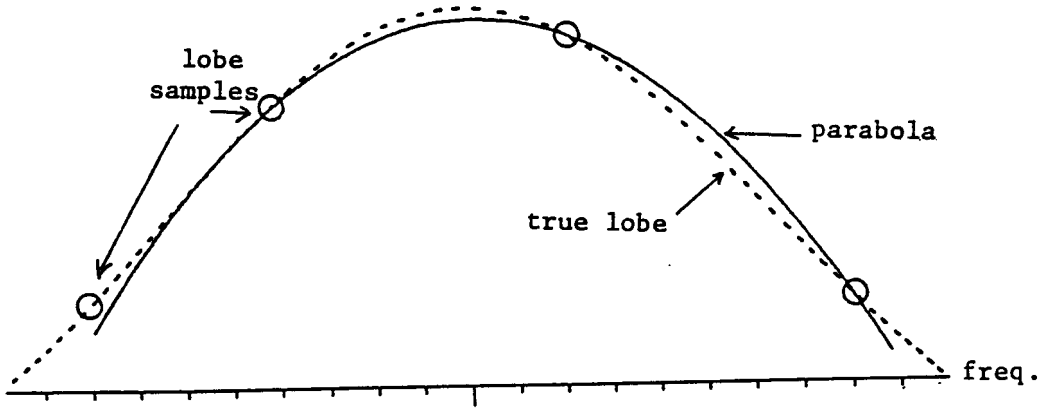

Figure 4: Parabolic interpolation of the highest three samples of a peak.

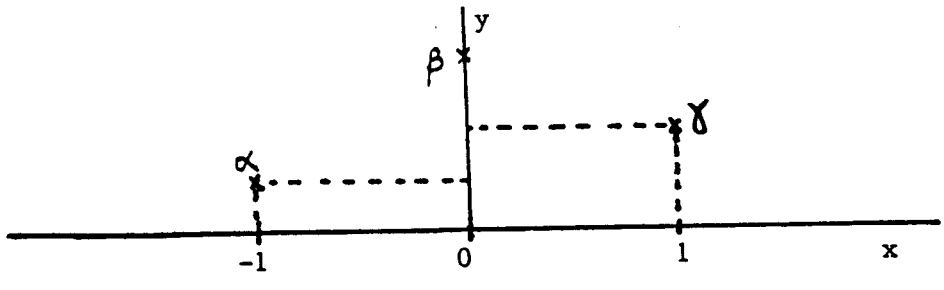

Figure 5: Coordinate system for the parabolic interpolation.

for all $k \neq k_\beta$. An example is shown in Figure 4. We desire a general parabola of the form

$$y(x) \stackrel{\Delta}{=} a(x-p)^2 + b$$

such that $y(-1) = \alpha$, $y(0) = \beta$, and $y(1) = \gamma$, where α, β, and γ are the values of the three highest samples:

$$\alpha \stackrel{\Delta}{=} 20\log_{10}\left|\tilde{x}'_m(e^{j\omega_{k_\beta-1}})\right| \tag{12}$$

$$\beta \stackrel{\Delta}{=} 20\log_{10}\left|\tilde{x}'_m(e^{j\omega_{k_\beta}})\right| \tag{13}$$

$$\gamma \stackrel{\Delta}{=} 20\log_{10}\left|\tilde{x}'_m(e^{j\omega_{k_\beta+1}})\right| \tag{14}$$

We have found empirically that the frequencies tend to be about twice as accurate when dB magnitude is used rather than just linear magnitude. An interesting open question is what is the optimum nonlinear compression of the magnitude spectrum when quadratically interpolating it to estimate peak locations.

Solving for the parabola peak location p, we get

$$p = \frac{1}{2}\frac{\alpha-\gamma}{\alpha-2\beta+\gamma}$$

and the estimate of the true peak location (in bins) will be

$$k^* \stackrel{\Delta}{=} k_\beta + p$$

and the peak frequency in Hz is $f_s k^*/N$. Using p, the peak height estimate is then

$$y(p) = \beta - \frac{1}{4}(\alpha-\gamma)p$$

The magnitude spectrum is used to find p, but $y(p)$ can be computed separately for the real and imaginary parts of the complex spectrum to yield a complex-valued peak estimate (magnitude and phase).

Once an interpolated peak location has been found, the entire local maximum in the spectrum is removed. This allows the same algorithm to be used for the next peak. This peak detection and deletion process is continued until the maximum number of peaks specified by the user is found.

6 Peak Matching (Step 5)

The peak detection process returns the prominent peaks in a given frame sorted by frequency. The next step is to assign some subset of these peaks to oscillator trajectories, which is done by the peak matching algorithm. If the number of spectral peaks were constant with slowly changing amplitudes and frequencies along the sound, this task would be straightforward. However, it is not always immediately obvious how to connect the spectral peaks of one frame with those of the next.

To describe the peak matching process, let's assume that the frequency tracks were initialized at frame 1 and we are currently at frame m. Suppose that at frame $m-1$ the frequency values for the p tracks are $f_1, f_2, \ldots, f_p$, and that we want to match them to the r peaks, with frequencies $g_1, g_2, \ldots, g_r$, of frame m.

Each track looks for its peak in frame m by finding the one which is closest in frequency to its current value. The ith track claims frequency g_j for which $|f_i - g_j|$ is minimum. The change in frequency must be less than a specified maximum $\Delta(f_i)$, which can be a frequency-dependent limit (e.g., linear, corresponding to a relative frequency change limit). The possible situations are as follows:

(1) If a match is found inside the maximum change limit, the track is continued (unless there is a conflict to resolve, as described below).

(2) If no match is made, it is assumed that the track with frequency f_i must "turn off" entering frame m, and f_i is matched to itself with zero magnitude. Since oscillator amplitudes are linearly ramped from one the frame to the next, the terminating track will ramp to zero over the duration of one frame hop. This track will still exist (at zero amplitude), and if it ever finds a frame with a spectral peak within its capture range $\Delta(f_i)$, it will "turned back on," ramping its amplitude up to the newly detected value. It is sometimes necessary to introduce some hysteresis into the turning on and off process in order to prevent "burbling" of the tracks whose peaks sometimes make the cut and sometimes don't. Normally this problem can be avoided by searching for many more spectral peaks than there are oscillators to allocate.

(3) If a track finds a match which has already been claimed by another track, we give the peak to the track which is closest in frequency. and the "losing" looks for another match. If the current track loses the conflict, it simply picks the best available non-conflicting peak. If the current track wins the conflict, it calls the assignment procedure recursively on behalf of the dislodged track. When the dislodged track finds the same peak and wants to claim it, it will see there is a conflict which it loses and will move on. This process is repeated for each track, solving conflicts recursively, until all existing tracks are matched or "turned-off".

After each track has extended itself forward in time or turned off, the peaks of frame m which have not been used are considered to be new trajectories and a new track is "started-up" for each one of them up to the maximum number of oscillators specified (which again should be less than the number of spectral peaks detected). The new oscillator tracks are started at frame $n-1$ with zero magnitude and ramp to the correct amplitude at the current frame m.

Once the program has finished, the peak trajectories for a sound look as in Fig. 6.

7 Parameter Modifications (Step 6)

The possibilities that STFT techniques offer for modifying the analysis results before resynthesis have an enormous number of musical applications. Quatieri and McAulay [20] give a good discussion of some useful modifications for speech applications. By scaling and/or resampling the amplitude and the frequency trajectories, a host of sound transformations can be accomplished.

Time-scale modifications can be accomplished by resampling the amplitude, frequency, and phase trajectories. This can be done simply by changing the hop size R in the resynthesis (although for best results the hop size should change adaptively, avoiding time-scale modifications during voice consonants or attacks, for example). This has the effect of slowing down or speeding up the sound while maintaining pitch and formant structure. Obviously this can also be done for a time-varying modification by having a time-varying hop size R. However, due to the sinusoidal representation, when a considerable time stretch is done in a "noisy" part of a sound, the individual sinewaves start to be heard and the noise-like quality is lost.

Frequency transformations, with or without time scaling, are also possible. A simple one is

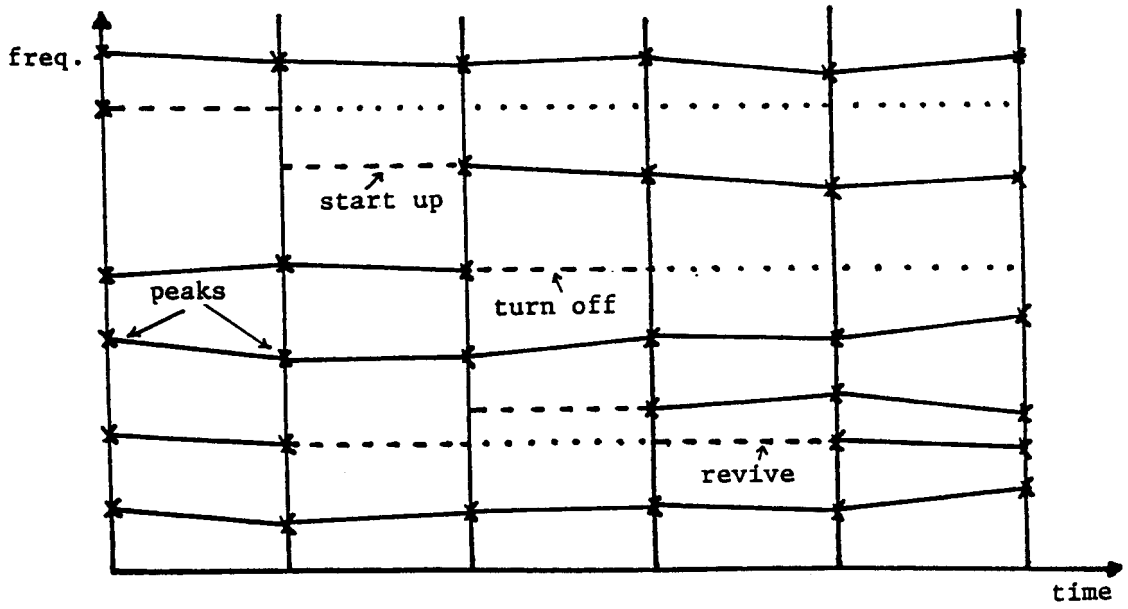

Figure 6: Peak trajectories for a piano tone.

to scale the frequencies to alter pitch and formant structure together. A more powerful class of spectral modifications comes about by decoupling the sinusoidal frequencies (which convey pitch and inharmonicity information) from the spectral envelope (which conveys formant structure so important to speech perception and timbre). By measuring the formant envelope of a harmonic spectrum (e.g., by drawing straight lines or splines across the tops of the sinusoidal peaks in the spectrum and then smoothing), modifications can be introduced which only alter the pitch or only alter the formants. Other ways to measure formant envelopes include cepstral smoothing [15] and the fitting of low-order LPC models to the inverse FFT of the squared magnitude of the spectrum [9]. By modulating the flattened (by dividing out the formant envelope) spectrum of one sound by the formant-envelope of a second sound, "cross-synthesis" is obtained. Much more complex modifications are possible.

Not all spectral modifications are "legal," however. As mentioned earlier, multiplicative modifications (simple filtering, equalization, etc.) are straightforward; we simply zero-pad sufficiently to accomodate spreading in time due to convolution. It is also possible to approximate nonlinear functions of the spectrum in terms of polynomial expansions (which are purely multiplicative). When using data derived filters, such as measured formant envelopes, it is a good idea to smooth the spectral envelopes sufficiently that their inverse FFT is shorter in duration than the amount of zero-padding provided. One way to monitor time-aliasing distortion is to measure the signal energy at the midpoint of the inverse-FFT output buffer, relative to the total energy in the buffer, just before adding it to the final outgoing overlap-add reconstruction; little relative energy in the "maximum-positive" and "minimum negative" time regions indicates little time aliasing. The general problem to avoid here is drastic spectral modifications which correspond to long filters in the time domain for which insufficient zero-padding has been provided. An inverse FFT of the spectral modification function will show its time duration and indicate zero-padding requirements.

The general rule (worth remembering in any audio filtering context) is "be gentle in the frequency domain."

8 Synthesis (Step 7)

The analysis portion of PARSHL returns a set of amplitudes $\hat{A}^m$, frequencies $\hat{\omega}^m$, and phases $\hat{\theta}^m$, for each frame index m, with a "triad" $(\hat{A}_r^m, \hat{\omega}_r^m, \hat{\theta}_r^m)$ for each track r. From this analysis data the program has the option of generating a synthetic sound.

The synthesis is done one frame at a time. The frame at hop m, specifies the synthesis buffer

$$s^m(n) = \sum_{r=1}^{R^m} \hat{A}_r^m \cos[n\hat{\omega}_r^m + \hat{\theta}_r^m]$$

where R^m is the number of tracks present at frame m; $m = 0, 1, 2, \ldots, S-1$; and S is the length of the synthesis buffer (without any time scaling $S = R$, the analysis hop size). To avoid "clicks" at the frame boundaries, the parameters $(\hat{A}_r^m, \hat{\omega}_r^m, \hat{\theta}_r^m)$ are smoothly interpolated from frame to frame.

The parameter interpolation across time used in PARSHL is the same as that used by McAulay and Quatieri [12]. Let $(\hat{A}_r^{(m-1)}, \hat{\omega}_r^{(m-1)}, \hat{\theta}_r^{(m-1)})$ and $(\hat{A}_r^m, \hat{\omega}_r^m, \hat{\theta}_r^m)$ denote the sets of parameters at frames $m-1$ and m for the rth frequency track. They are taken to represent the state of the signal at time 0 (the left endpoint) of the frame.

The instantaneous amplitude $\hat{A}(n)$ is easily obtained by linear interpolation,

$$\hat{A}(n) = \hat{A}^{m-1} + \frac{(\hat{A}^m - \hat{A}^{m-1})}{S} n$$

where $n = 0, 1, \ldots, S-1$ is the time sample into the mth frame.

Frequency and phase values are tied together (frequency is the phase derivative), and they both control the instantaneous phase $\hat{\theta}(n)$. Given that four variables are affecting the instantaneous phase: $\hat{\omega}^{(m-1)}, \hat{\theta}^{(m-1)}, \hat{\omega}^m$, and $\hat{\theta}^m$, we need at least three degrees of freedom for its control, while linear interpolation only gives one. Therefore, we need at least a cubic polynomial as interpolation function, of the form

$$\hat{\theta}(n) = \zeta + \gamma n + \alpha n^2 + \beta n^3.$$

We will not go into the details of solving this equation since McAulay and Quatieri [12] go through every step. We will simply state the result:

$$\hat{\theta}(n) = \hat{\theta}^{(m-1)} + \hat{\omega}^{(m-1)} n + \alpha n^2 + \beta n^3$$

where α and β can be calculated using the end conditions at the frame boundaries,

$$\alpha = \frac{3}{S^2}(\hat{\theta}^m - \hat{\theta}^{m-1} - \hat{\omega}^{m-1} S + 2\pi M) - \frac{1}{S}(\hat{\omega}^m - \hat{\omega}^{m-1}) \quad (15)$$

$$\beta = \frac{-2}{S^3}(\hat{\theta}^m - \hat{\theta}^{m-1} - \hat{\omega}^{m-1} S + 2\pi M) + \frac{1}{S^2}(\hat{\omega}^m - \hat{\omega}^{m-1}) \quad (16)$$

This will give a set of interpolating functions depending on the value of M, among which we have to select the "maximally smooth" one. This can be done by choosing M to be the integer closest to x, where x is

$$x = \frac{1}{2\pi}\left[(\hat{\theta}^{m-1} - \hat{\omega}^{m-1} S - \hat{\theta}^m) + (\hat{\omega}^m - \hat{\omega}^{m+1})\frac{S}{2}\right]$$

and finally, the synthesis equation turns into

$$s^m(n) = \sum_{r=1}^{R^m} \hat{A}_r^m(n) \cos[\hat{\theta}_r^m(n)]$$

which smoothly goes from frame to frame and where each sinusoid accounts for both the rapid phase changes (frequency) and the slowly varying phase changes.

Figure 7 shows the result of the analysis/synthesis process using phase information and applied to a piano tone.

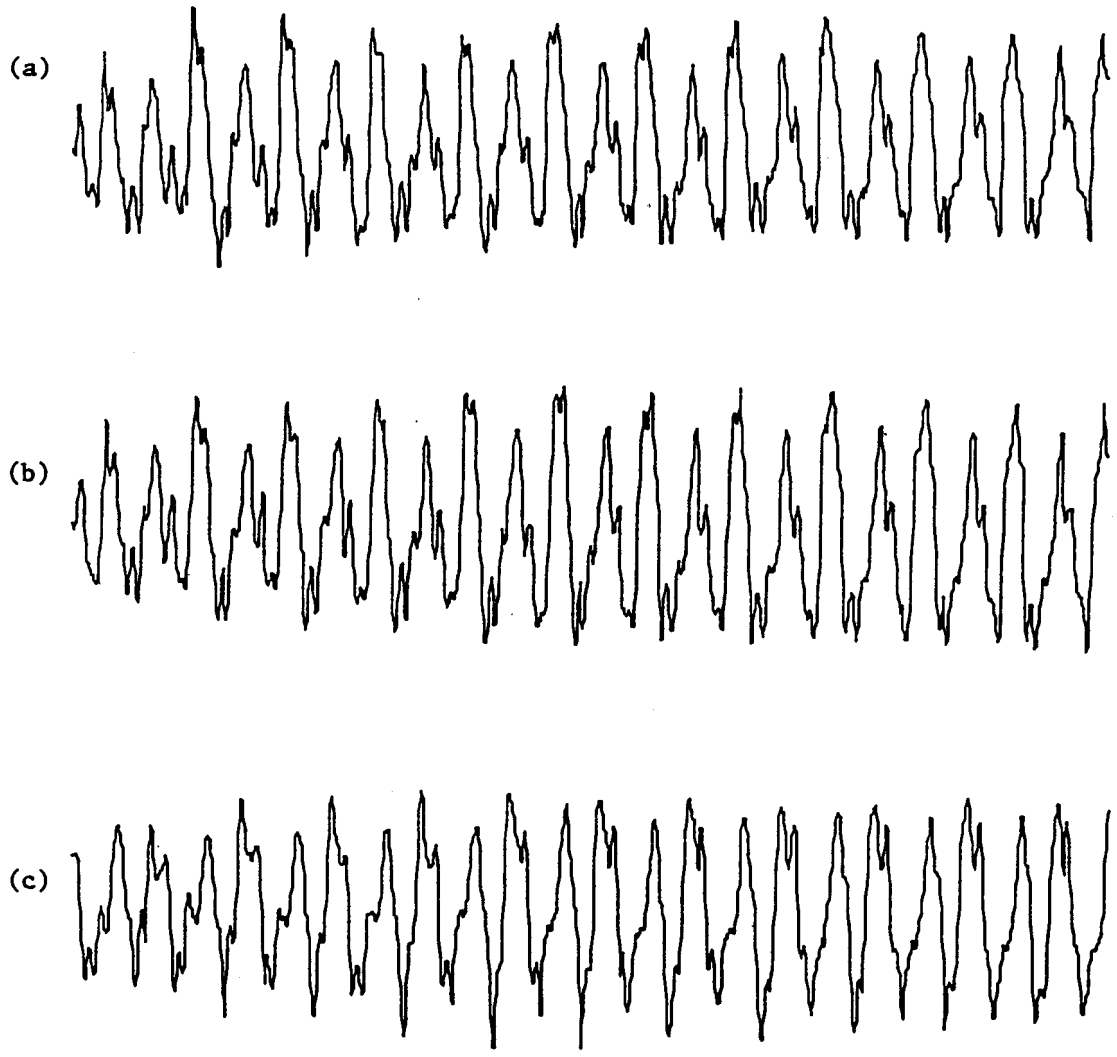

Figure 7: (a) Original piano tone, (b) synthesis with phase information, (c) synthesis without phase information.

9 Magnitude-only Analysis/Synthesis

A traditional result of sound perception is that the ear is sensitive principally to the short-time spectral magnitude and not to the phase, provided phase continuity is maintained. Our experience has been that this may or may not be true depending on the application, and in §11 we will discuss it. Obviously if the phase information is discarded, the analysis, the modification, and the resynthesis processes are simplified enormously. Thus we will use the magnitude-only option of the program whenever the application allows it.

In the peak detection process we calculate the magnitude and phase of each peak by using the complex spectrum. Once we decide to discard the phase information there is no need for complex spectra and we simply can calculate the magnitude of the peak by doing the parabolic interpolation directly on the log magnitude spectrum.

The synthesis also becomes easier; there is no need for a cubic function to interpolate the instantaneous phase. The phase will be a function of the instantaneous frequency and the only condition is phase continuity at the frame boundaries. Therefore, the frequency can be linearly interpolated from frame to frame, like the amplitude. Without phase matching the synthesized waveform will look very different from the original (Fig. 7), but the sound quality for many applications will be perceptually the same.

10 Preprocessing

The task of the program can be simplified and the analysis/synthesis results improved if the sound input is appropiately manipulated before running the program.

Most important is to equalize the input signal. This controls what it means to find spectral peaks in order of decreasing magnitude. Equalization can be accomplished in many ways and here we present some alternatives.

(1) A good equalization strategy for audio applications is to weight the incoming spectrum by the inverse of the equal-loudness contour for hearing at some nominal listening level (e.g. 60dB). This makes spectral magnitude ordering closer to perceptual audibility ordering.

(2) For more analytical work, the spectrum can be equalized to provide all partials at nearly the same amplitude (e.g., the asymptotic roll-off of all natural spectra can be eliminated). In this case, the peak finder is most likely to find and track all of the partials.

(3) A good equalization for noise-reduction applications is to "flatten" the noise floor. This option is useful when it is desired to set a fixed (frequency-independent) track rejection threshold just above the noise level.

(4) A fourth option is to perform adaptive equalization of types (2) or (3) above. That is, equalize each spectrum independently, or compute the equalization as a function of a weighted average of the most recent power spectrum (FFT squared magnitude) estimates.

Apart from equalization, another preprocessing strategy which has proven very useful is to reverse the sound in time. The attack of most sounds is quite "noisy" and PARSHL has a hard time finding the relevant partials in such a complex spectrum. Once the sound is reversed the program will encounter the end of the sound first, and since in most instrumental sounds this is a very stable part, the program will find a very clear definition of the partials. When the program gets to the sound attack, it will already be tracking the main partials. Since PARSHL has a fixed number of oscillators which can be allocated to discovered tracks, and since each track which disappears

removes its associated oscillator from the scene forever,[2] analyzing the sound tail to head tends to allocate the oscillators to the most important frequency tracks first.

11 Applications

The simplest application of PARSHL is as an analysis tool since we can get a very good picture of the evolution of the sound in time by looking at the amplitude, frequency and phase trajectories. The tracking characteristics of the technique yield more accurate amplitudes and frequencies than if the analysis were done with an equally spaced bank of filters (the traditional STFT implementation).

In speech applications, the most common use of the STFT is for data-reduction. With a set of amplitude, frequency and phase functions we can get a very accurate resynthesis of many sounds with much less information than for the original sampled sounds. From our work it is still not clear how important is the phase information in the case of resynthesis without modifications, but McAulay and Quatieri [12] have shown the importance of phase in the case of speech resynthesis.

One of the most interesting musical applications of the STFT techniques are given by their ability to separate temporal from spectral information, and, within each spectrum, pitch and harmonicity from formant information. In §7, Parameter Modifications, we discussed some of them, such as time scaling and pitch transposition. But this group of applications has a lot of possibilities that still need to be carefully explored. From the few experiments we have done to date, the tools presented give good results in situations where less flexible implementations do not, namely, when the input sound has inharmonic spectra and/or rapid frequency changes.

The main characteristic that differentiates this model from the traditional ones is the selectivity of spectral information and the phase tracking. This opens up new applications that are worth our attention. One of them is the use of additive synthesis in conjunction with other synthesis techniques. Since the program allows tracking of specific spectral components of a sound, we have the flexibility of synthesizing only part of a sound with additive, synthesis, leaving the rest for some other technique. For example, Serra [22] has used this program in conjunction with LPC techniques to model bar percussion instruments, and Marks and Polito [11] have modeled piano tones by using it in conjunction with FM synthesis. David Jaffe has had good success with birdsong, and Rachel Boughton used PARSHL to create abstractions of ocean sounds.

One of the problems encountered when using several techniques to synthesize the same sound is the difficulty of creating the perceptual fusion of the two synthesis components. By using phase information we have the possibility of matching the phases of the additive synthesis part to the rest of the sound (independently of what technique was used to generate it). This provides improved signal "splicing" capability, allowing very fast cross-fades (e.g., over one frame period).

PARSHL was originally written to properly analyze the steady state of piano sounds; it did not address modeling the attack of the piano sound for purposes of resynthesis. The phase tracking was primarily motivated by the idea of splicing the real attack (sampled waveform) to its synthesized steady state. It is well known that additive synthesis techniques have a very hard time synthesizing attacks, both due to their fast transition and their "noisy" characteristics. The problem is made more difficult by the fact that we are very sensitive to the quality of a sound's attack. For plucked or struck strings, if we are able to splice two or three periods, or a few milliseconds, of the original

[2] We tried reusing turned-off oscillators but found them to be more trouble than they were worth in our environment.

sound into our synthesized version the quality can improve considerably, retaining a large data-reduction factor and the possibility of manipulating the synthesis part. When this is attempted without the phase information, the splice, even if we do a smooth cross-fade over a number of samples, can be very noticeable. By simply adding the phase data the task becomes comparatively easy, and the splice is much closer to inaudible.

12 Conclusions

In this paper an analysis/synthesis technique based on a sinusoidal representation was presented that has proven to be very appropriate for signals which are well characterized as a sum of inharmonic sinusoids with slowly varying amplitudes and frequencies. The previously used harmonic vocoder techniques have been relatively unwieldy in the inharmonic case, and less robust even in the harmonic case. PARSHL obtains the sinusoidal representation of the input sound by tracking the amplitude, frequency, and phase of the most prominent peaks in a series of spectra computed using the Fast Fourier Transform of successive, overlapping, windowed data frames, taken over the duration of a sound. We have mentioned some of the musical applications of this sinusoidal representation.

Continuing the work with this analysis/synthesis technique we are implementing PARSHL on a Lisp Machine with an attached FPS AP120B array processor. We plan to study further its sound transformation possibilities and the use of PARSHL in conjunction with other analysis/synthesis techniques such as Linear Predictive Coding (LPC) [10].

The basic "FFT processor" at the heart of PARSHL provides a ready point of departure for many other STFT applications such as FIR filtering, speech coding, noise reduction, adaptive equalization, cross-synthesis, and many more. The basic parameter trade-offs discussed in this paper are universal across all of these applications.

Although PARSHL was designed to analyze piano recordings, it has proven very successful in extracting additive synthesis parameters for radically inharmonic sounds. It provides interesting effects when made to extract peak trajectories in signals which are not describable as sums of sinusoids (such as noise or ocean recordings). PARSHL has even demonstrated that speech can be intelligible after reducing it to only the three strongest sinusoidal components.

The surprising success of additive synthesis from spectral peaks suggests a close connection with audio perception. Perhaps timbre perception is based on data reduction in the brain similar to that carried out by PARSHL. This data reduction goes beyond what is provided by critical-band masking. Perhaps a higher-level theory of "timbral masking" or "main feature dominance" is appropriate, wherein the principal spectral features serve to define the timbre, masking lower-level (though unmasked) structure. The lower-level features would have to be restricted to qualitatively similar behavior in order that they be "implied" by the louder features. Another point of view is that the spectral peaks are analogous to the outlines of figures in a picture—they capture enough of the perceptual cues to trigger the proper percept; memory itself may then serve to fill in the implied spectral features (at least for a time).

Techniques such as PARSHL provide a powerful analysis tool toward extracting signal parameters matched to the characteristics of hearing. Such an approach is perhaps the best single way to obtain cost-effective, analysis-based synthesis of any sound.

Acknowledgments

We wish to thank Dynacord, Inc., for supporting the development of the first version of PARSHL in the summer of 1985. We also wish to acknowledge the valuable contributions of Gerold Schrutz (Dynacord) during that time.

Software Listing

The online version[3] of this paper contains a complete code listing for the original PARSHL program.

References

[1] J. B. Allen, "Short term spectral analysis, synthesis, and modification by discrete Fourier transform," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-25, pp. 235–238, June 1977.

[2] J. B. Allen, "Application of the short-time Fourier transform to speech processing and spectral analysis," *Proc. IEEE ICASSP-82*, pp. 1012–1015, 1982.

[3] J. B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, pp. 1558–1564, Nov. 1977.

[4] H. Chamberlin, *Musical Applications of Microprocessors*, New Jersey: Hayden Book Co., Inc., 1980.

[5] R. Crochiere, "A weighted overlap-add method of short-time Fourier analysis/synthesis," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-28, pp. 99–102, Feb 1980.

[6] M. Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, pp. 14–27, Winter 1986.

[7] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, pp. 51–83, Jan 1978.

[8] J. F. Kaiser, "Using the I_0 – sinh window function," *IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications*, pp. 20–23, April 22–25 1974, Reprinted in [**?**], pp. 123–126.

[9] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, April 1975.

[10] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, New York: Springer Verlag, 1976.

[11] J. Marks and J. Polito, "Modeling piano tones," in *Proceedings of the 1986 International Computer Music Conference, The Hague*, Computer Music Association, 1986.

[3] http://www-ccrma.stanford.edu/~jos/parshl/

[12] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-34, pp. 744–754, Aug 1986.

[13] J. A. Moorer, "The use of the phase vocoder in computer music applications," *Journal of the Audio Engineering Society*, vol. 26, pp. 42–45, Jan./Feb. 1978.

[14] A. H. Nuttall, "Some windows with very good sidelobe behavior," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-29, pp. 84–91, Feb 1981.

[15] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

[16] A. Papoulis, *Signal Analysis*, New York: McGraw-Hill, 1977.

[17] M. R. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-24, pp. 243–248, June 1976.

[18] M. R. Portnoff, "Time–frequency representation of digital signals and systems based on short–time Fourier analysis," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-28, pp. 55–69, Feb 1980.

[19] R. Portnoff, "Short-time Fourier analysis of sampled speech," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. 29, no. 3, pp. 364–373, 1981.

[20] T. F. Quatieri and R. J. McAulay, "Speech transformations based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-34, pp. 1449–1464, Dec 1986.

[21] X. Rodet and P. Depalle, "Spectral envelopes and inverse FFT synthesis," *Proc. 93rd Convention of the Audio Engineering Society, San Francisco*, 1992, Preprint 3393 (H-3).

[22] X. Serra, "A computer model for bar percussion instruments," in *Proceedings of the 1986 International Computer Music Conference, The Hague*, pp. 257–262, Computer Music Association, 1986.

[23] M. J. T. Smith and T. P. Barnwell, "A unifying framework for analysis/synthesis systems based on maximally decimated filter banks," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Tampa, Florida*, (New York), pp. 521–524, IEEE Press, 1985.

[24] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1993.