TIME-STRETCHING USING THE INSTANTANEOUS FREQUENCY DISTRIBUTION AND PARTIAL TRACKING

Victor Lazzarini
Department of Music
NUI, Maynooth
Ireland
Email:
Victor.Lazzarini@nuim.ie

Joe Timoney
Dept. of Computer Science
NUI, Maynooth
Ireland
Email:
Joe.Timoney@cs.nuim.ie

Tom Lysaght
Dept. of Computer Science
NUI, Maynooth
Ireland
Email:
Tom.Lysaght@cs.nuim.ie

ABSTRACT

This article presents a method of signal timescale modification using spectral analysis-resynthesis. It discusses an alternative technique for instantaneous frequency estimation, the Instantaneous Frequency Distribution (IFD). The partial tracking analysis employed in this process is explained in some detail, followed by a look into the resynthesis method. The article discusses this technique of time-stretching in comparison to the standard phase vocoder process. Performance details and specific aspects of this implementation are examined, including the C++ code for a time-stretching application.

1. INTRODUCTION

Time-stretching is a transformation process frequently used in many computer music applications. It involves the change in the duration, or time-scaling, of a signal independent of its frequency scale. A variety of techniques exist for its implementation, in both time and spectral domains. The quality of the time-stretched output varies quite a lot between these different methods. Many of them will introduce a certain amount of artifacts, which sometimes may be objectionable.

A widespread method of time-stretching using spectral analysis-resynthesis is the phase vocoder[2]. The process relies on the use of the Short-Time Fourier Transform (STFT), followed by polar conversion and instantaneous frequency estimation. The analysed data can then be resynthesised at a different time rate using either an inverse process of phase integration, followed by rectangular conversion and the inverse transform, or by additive synthesis. Depending on the source signal and on the time-stretching amount, the phase vocoder will produce an acceptable result. However, due to many inherent problems of phase recuperation involved in the process[9], artifacts will be produced, which in some cases might result in a low-quality output.

This article describes an alternative method of time-stretching based on spectral analysis-resynthesis. This technique employs a superior method of instantaneous frequency estimation, followed by partial tracking analysis and additive resynthesis. With this method is possible to provide an enhanced quality time-stretching, without the problems associated with more traditional approaches.

2. THE INSTANTANEOUS FREQUENCY DISTRIBUTION

The method of frequency estimation used in this work is given by the instantaneous frequency distribution (IFD) algorithm. This was proposed, independently, by Friedman[3] and Toshihiko Abe[1]. It uses some of the principles also seen in the phase vocoder algorithm, but its mathematical formulation is more complex. The basic idea is that the instantaneous frequency detected at a certain band is the time derivative of the phase. Using Euler's relationship, we can define the output of the Discrete Fourier Transform (DFT) in polar form. This is shown below, using $\omega = 2\pi k/N$:

$$DFT(x(n),k,t) = R(\omega,t) \times e^{j\theta(\omega,t)} \quad (1)$$

The phase detected by band k at time-point t is $\theta(2\pi kn/N,\ t)$ and the magnitude is $R(2\pi kn/N,\ t)$. The Instantaneous Frequency Distribution of $x(n)$ at time t is then the time derivative of the STFT phase output:

$$IFD(x(n),k,t) = \frac{\partial}{\partial t}\theta(\omega,t) \quad (2)$$

This can be intuitively understood as the measurement of the *rate of rotation* of the phase of a sinusoidal signal. In the phase vocoder, this is estimated by crudely taking the difference between phase values in successive frames. The IFD actually calculates the time derivative of the phase directly, from data corresponding to a single time-point. We will start by using the STFT, taken as a series of DFTs of a rotated and windowed input signal:

$$\begin{aligned} &STFT(x(n),k,t) = \\ &= e^{-j2\pi kt/N}\ \frac{1}{N}\sum_{m=0}^{N-1} w(m)x(m+t)e^{-j2\pi km/N} \\ &= e^{-j2\pi kt/N} DFT(x_t(m),k) \end{aligned} \quad (3)$$

Each DFT is taken from $x_t(m) = w(m)x(m+t)$, which is the windowed input signal at time-point t. The multiplication by the complex exponential can be done in the time domain as a rotation of the input. Now the phase can be isolated from the magnitude spectrum. This is done by first taking the logarithm of the DFT output in polar form. From (1), we have:

$$\ln[DFT(x_t(m),k)] = \ln[R(\omega,t)] + j\theta(\omega,t) \quad (4)$$

It is clear from the above that the phase is the imaginary part of the logarithm of the DFT, *imag{ln(DFT)}*. Now the derivative of the phase can be expressed in terms of the DFT of the signal:

$$\frac{\partial}{\partial t}\theta(\omega,t) = \frac{\partial}{\partial t} imag\left\{\ln[DFT(x_t(m),k)]\right\} = \\ = imag\left\{\frac{1}{DFT(x_t(m),k)} \times \frac{\partial}{\partial t} DFT(x_t(m),k)\right\} \quad (5)$$

All that is necessary is to find the derivative of the DFT. This can be done by changing the summation variable of the transform, using $r = m+ t$, so that the only function of the time variable t left inside the summation is the window:

$$DFT(x_t(m),k) = \frac{1}{N}\sum_{m=0}^{N-1} x(m+t)w(m)e^{-j\omega m} = \\ = \frac{1}{N}\sum_{r=0}^{N-1} x(r)w(r-t)e^{-j\omega(r-t)} = \quad (6) \\ = e^{j\omega t}\frac{1}{N}\sum_{r=0}^{N-1} x(r)w(r-t)e^{-j\omega r}$$

Now it is a simple matter of taking the derivative of the above product. As the derivative of the complex exponential is trivial, we are left only with the derivative of the window function:

$$\frac{\partial}{\partial t} DFT(x_t(m),k) = \frac{\partial}{\partial t}\left\{e^{j\omega t}\frac{1}{N}\sum_{r=0}^{N-1} x(r)w(r-t)e^{-j\omega r}\right\} = \\ = j\omega e^{j\omega t}\frac{1}{N}\sum_{r=0}^{N-1} x(r)w(r-t)e^{-j\omega r} \quad (7) \\ + e^{j\omega t}\frac{1}{N}\sum_{r=0}^{N-1} x(r)\left\{\frac{\partial}{\partial t} w(r-t)\right\}e^{-j\omega r}$$

Reverting to the previous formulation of the DFT (using the converse of (6), with $m = r - t$):

$$\frac{\partial}{\partial t} DFT(x_t(m),k) = \frac{1}{N}\sum_{r=0}^{N-1} x(m+t)w(m)e^{-j\omega m} \\ + \frac{1}{N}\sum_{r=0}^{N-1} x(m+t)\left\{\frac{\partial}{\partial(r-m)} w(m)\right\}e^{-j\omega m} = \\ = j\omega DFT(x_t(m),k) + DFT(x'_t(m),k) \quad (8) \\ (with \quad x'_t(m) = -\frac{\partial}{\partial m} w(m)x(m+t))$$

Substituting back in (5) and using the definition of the IFD given in (2), the final formulation is obtained:

$$IFD(x(n),k,t) = \omega + imag\left\{\frac{DFT(x'_t(m),k)}{DFT(x_t(m),k)}\right\} \quad (9)$$

As demonstrated, the IFD is formulated as a ratio of two DFTs, one taken from a windowed signal and the other using the same signal windowed by the negative derivative of the same analysis window. The derivative of the analysis window is generated by computing the differences between its consecutive samples. Re-arranging the formula, it is possible to see that the instantaneous frequency deviation *d(x(n),k,t)* from the bin centre frequencies is inversely proportional to the signal power at that bin (eq. 10). This follows from the fact that the closer a frequency peak is to the bin centre, the higher the bin magnitude will be.

$$d(x(n),k,t) = imag\left\{\frac{DFT(x'_t(m),k)}{DFT(x_t(m),k)}\right\} = \\ = \frac{imag\left[DFT^*(x_t(m),k)DFT(x'_t(m),k)\right]}{\left|DFT(x_t(m),k)\right|^2} \quad (10)$$

The method of spectral analysis employed here will use the modulus of the DFT of a windowed frame to obtain the magnitudes. At the same time, the IFD will be used to estimate the frequencies. It is important to point out that we are estimating the frequency using a single analysis time-point. As far as the instantaneous frequency estimation is concerned, the amount of analysis frame overlap is of no consequence. The IFD formulation shares some common aspects with the hopsize-1 phase vocoder described in [8], but provides a much more accurate result, as outlined by Hainsworth and McLeod [4]. In a recent study, this method has been shown to provide a superior frequency analysis to a number of other methods, including the standard phase vocoder [5].

3. PARTIAL TRACKING

Using the IFD and magnitudes from the analysis method described above, it is possible to employ a peak-picking algorithm to isolate partials. These can be organised into spectral tracks, containing frequency and amplitude information. Similar partial-tracking methods are employed in sinusoidal modelling of speech [7] and of arbitrary audio/music signals [11][12], with deterministic/stochastic component separation. Both methods also retain and use phase spectra from STFT analysis. In this work, however, we are only concerned with amplitudes and instantaneous frequencies. Also, the tracking method here does not attempt to separate deterministic and stochastic components, but will keep them together in the analysis data.

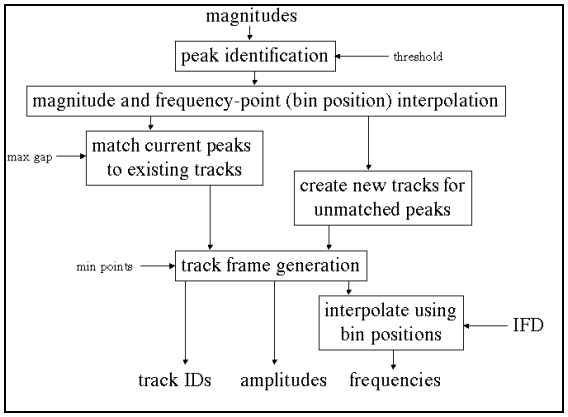

Figure 1. Partial track analysis

The principle behind the partial track analysis is very simple, although its implementation is somewhat involved.

Using the magnitudes, spectral peaks at integral frequency points (bins) will be identified. A thresholding mechanism is used to eliminate low-amplitude components. The exact peak position and amplitude can be estimated using a quadratic interpolation procedure based on the magnitudes of the bins around the peaks. With the interpolated frequency points, it is possible to find the exact values for the frequencies obtained originally from the IFD input, through linear interpolation. These will then, together with the amplitude, form a 'track', linked to each detected peak.

The track will only exist as such if there is some consistency in consecutive frames, ie. if there is some matching between peaks found at each time-point. When peaks are short-lived, they will not make a track. Conversely, when a peak disappears, we will have to wait a few frames to declare the track as finished. Most of the process involved in this analysis becomes one of track management, which accounts for the more complex aspects of the algorithm.

The track generation algorithm is shown in fig. 1. It takes in the DFT magnitudes, a threshold, maximum gap (in time-points) between peaks in a track and minimum number of time-points for a track, as well as the instantaneous frequencies of each bin, from the IFD.

The present implementation of the partial analysis algorithm can be described as a streaming process. Track frames are generated for every hop size of the analysed signal. Here, unlike other more straightforward implementations, only a single signal frame is present, or known, at a given time by the analysis algorithm, making it suitable, for instance, for realtime processing. For such implementation a number of special details are required. For instance, a record of past peaks is needed for the peak matching operation. Also, if the required minimum number of time-points exceeds one, a delay will be introduced between the input and the output of the analysis.

4. RESYNTHESIS

As discussed above, the partial tracking analysis will output tracks made up of frequencies and amplitudes. This in turn can be used for additive re-synthesis of the signal or of portions of its spectrum. A typical method involves the use of the frequency parameter to calculate the varying phase used to drive an oscillator and the amplitude to scale its output. The parameters can be interpolated linearly on a frame-by-frame basis, which was found to be efficient and sufficiently precise for many applications. The resynthesis process is described by eqs 11 to 14:

$$output(n) = \sum_{trk=1}^{N} a(trk,n)\cos(2\pi\theta(trk,n)/sr) \quad (11)$$

$$\theta(trk, n+1) = \theta(trk, n) + f(trk, n) \quad (12)$$

$$f(trk,n) = f(trk,t) + [f(trk,t+n) - f(trk,t)](n-t)/h \quad (13)$$

$$a(trk,n) = a(trk,t) + [a(trk,t+n) - a(trk,t)](n-t)/h \quad (14)$$

where *a(trk,n)* and *f(trk,n)* are the interpolated amplitudes and frequencies, *h* is the hopsize and *t* the analysis time-points.

Additive resynthesis uses the analysis track frames to drive a bank of cosine wave oscillators . In this streaming implementation, the process will use the track IDs to match tracks between analysis frames, in order to perform properly the interpolation of amplitude and frequency. High-quality frequency shifting, scaling or warping is also possible, by introducing a scaling control on each oscillator frequency.

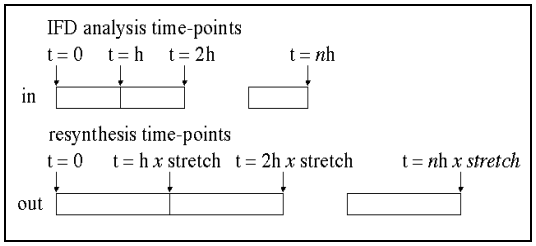

Figure 2. Time-stretching

5. TIME-STRETCHING

Time-stretching, independent of frequency, can be implemented with the processes described above. After IFD/magnitude and partial tracking, the resynthesis can be made to proceed at any time rate, since the track data contain basically the parametric values of amplitude and frequencies. By spacing the resynthesis time-points to different hopsizes (in eq. 12 to 14), time-scale modification (either stretching or compression) is produced (fig. 2).

The quality of the time-stretched output produced with this method seems to be much superior to other common techniques of timescale modification, particularly the phase vocoder. In fact, since this process is based on the analysis of spectral peaks, the typical artifacts associated with that technique, such as phasiness and modulation, are absent from the resynthesised sound. A number of comparative tests were run, with different sound sources, and it was found that the method described offers considerable the quality of the output, especially for larger time-stretching factors.

5.1. Implementation details

In this work, the IFD, partial analysis and resynthesis have been implemented as classes in the SndObj library [6][12]. The instantaneous frequency and magnitude analysis are generated by an IFGram object from a time-domain signal. Partial tracking is performed by a SinAnal object, which takes the IFGram output. Finally, resynthesis is performed by an object of the AdSyn class.

All processing objects are fed into a SndThread object, which manages the sound processing thread. This basic code, shown in fig 8, can be wrapped up in a GUI framework or used directly as a command-line application. The output object, shown here as using the SndWave class, can alternatively use the SndRTIO or SndASIO classes for realtime output in systems with such capability.

5.2. Other processes

In addition to time-stretching, the analysis-synthesis method described here has a variety of other potential applications, such as most of the ones described in [13]. In the partial tracking methods, for instance, basic noise-reduction can be performed by adjusting the analysis threshold parameter. Transients can be reduced by increasing the minimum number of analysis time points. Non-linear filtering can be achieved by limiting the number of analysis and/or resynthesis tracks. Other effects can be implemented by custom track processing classes. Furthermore, with the track analysis framework in place, it is possible to design and experiment new processes that take advantage of this data format

```
SndThread thread;
// cosine wavetable
HarmTable tb(10000,1,1, 0.25);
// hanning window
HammingTable win(ffts,0.5);

// SndObj chain
// dcm is analysis hopsize
// itp is resynthesis hopsize
// itp:dcm is timescale ratio
SndWave in(ifile,READ,1,16,0,0,dcm);
SndIn   ins(&in,1,dcm);
IFGram  ifd(&win,&ins,1.f,ffts,dcm);
SinAnal trks(&ifd,thrsh,intrks,1,3);
AdSyn   syn(&trks,outrks,&tb,1.f,scl,itp);
SndWave out(ofile,OVERWRITE,1,16,0,0,itp);
out.SetOutput(1,&syn);

// sound thread set-up
thread.AddObj(&ins);
thread.AddObj(&ifd);
thread.AddObj(&trks);
thread.AddObj(&syn);
thread.AddObj(&in,SNDIO_IN);
thread.AddObj(&out,SNDIO_OUT);

// processing
thread.ProcOn();
while(!input.Eof());
thread.ProcOff();
```

Figure 3. C++ code for a time-stretching application.

6. FUTURE PROSPECTS

The simple resynthesis method proposed in this paper has been successful for the resynthesis of arbitrary signals. However, it was found that further enhancements to the output sound quality can be achieved using better interpolation methods. One such method is the cubic interpolation algorithm proposed in [7], using the retained phases in resynthesis. However, the original formulation for this method does not allow for changes in the signal timescale. An adaptation of this method is currently being investigated for track resynthesis, as an alternative to the method presented here.

7. REFERENCES

[1] Abe T, et al. “The IF spectrogram: a new spectral representation,Ó*Proc. ASVA 97*, pp. 423-430, 1997.

[2] Dolson, M. “The phase vocoder tutorialÓ *Computer Music Journal, 10(4)*, pp. 14-27, 1986.

[3] Friedman, DH. “Instantaneous-frequency distribution vs time: an interpretation of the phase structure of speechÓ *Proc. ICASSP*, pp 1121-4, 1985.

[4] Hainsworth, S, Mclead, M. *Time-frequency reassignment: a review and analysis*. Technical Report, Cambridge Univ. Eng. Dept. CUED/FENG/TR.459, 2003.

[5] Keiler, F, Marchand, S. “Survey on extraction of sinusoids in stationary SoundsÓ *Proc. of DAFx02*, pp-217-221, 2002.

[6] Lazzarini, V. “The sound object libraryÓ *Organised Sound 5 (1).* pp. 35-49, 2000.

[7] McCaulay, RJ, Quatieri, TF. “Speech Analysis/Synthesis Based on a Sinusoidal RepresentationÓ *IEEE Trans. On Acoustics, Speech, and Signal Processing, ASSP-34 (4)*, 1986.

[8] Puckette, MS, Brown, JC. “Accuracy of frequency estimates using the phase vocoderÓ *IEE Transactions on Speech and Audio Processing 6 (2)*, pp. 166-177.

[9] Puckette, MS. “Phase-locked vocoderÓ *Proc. IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, 1995.

[10] Smith, J, Serra, X. “PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representationÓ*Proc. of the ICMC* 87, Tokio, 2002.

[11] Serra, X. “Musical Sound Modelling with Sinusoids plus NoiseÓ in: G.D. Poli et al (eds.), *Musical Signal Processing*, Swets & Zeitlinger Publishers, Amsterdam. 1997.

[12] Timoney, J, Lazzarini, V and Lysaght, T. “New SndObj classes for sinusoidal modellingÓ *Proc. of DAFx02*, pp-217-221, 2002.

[13] Verfaille, V, Depalle P. “Adaptive Effects Based on STFT, Using a Source-Filter Model". *Proc. of DAFx04*, pp. 296-301 2004.