2008-01-01

Time and Pitch Scale Modification: a Real-time Framework and Tutorial

Dan Barry
Dublin Institute of Technology, dan.barry@dit.ie

David Dorran
Dublin Institute of Technology, david.dorran@dit.ie

Eugene Coyle
Dublin Institute of Technology, Eugene.Coyle@dit.ie

Recommended Citation

Barry, D., Dorran, D. & Coyle, E. Time and pitch scale modification: a real-time framework and tutorial. Proceedings of the *11th. International Conferenc eon Digital Audio Effets (DAFx-08), Espoo, Finland, September 1-4, 2008.*

TIME AND PITCH SCALE MODIFICATION: A REAL-TIME FRAMEWORK AND TUTORIAL

Dan Barry
Audio Research Group,
Dublin Institute of Technology,
Dublin, Ireland
dan.barry@dit.ie

David Dorran
Audio Research Group,
Dublin Institute of Technology,
Dublin, Ireland
david.dorran@dit.ie

Eugene Coyle
Audio Research Group,
Dublin Institute of Technology,
Dublin, Ireland
eugene.coyle@dit.ie

ABSTRACT

A framework is presented which is designed to address the issues related to the real-time implementation of time-scale and pitch-scale modification algorithms. This framework can be used as the basis for the developments of applications which allow for a seamless real-time transition between continually varying time-scale and pitch-scale parameters which arise as a result of manual or automatic intervention.

1. INTRODUCTION

Time-scale modification algorithms enable the playback rate of audio content to be arbitrarily slowed down or speeded up without affecting the local pitch content of the audio signal. Time-scale modification (TSM) is typically used to change the tempo of musical audio content, and the playback rate of speech. Conversely, pitch-scale modification (PSM) algorithms enable pitch shifting without affecting the playback rate of the audio content. Typical uses include key transposition or harmonisation for musical audio content, and voice modification in speech based audio. A significant amount of research has been dedicated to both TSM and PSM yielding a variety of time and frequency domain algorithms. Additionally, several methods have been proposed and successfully implemented to deal with the known artefacts and shortcomings of the fundamental approaches to TSM. Despite this abundance of literature and commercial applications readily available, there is still a lack of information, understanding and consideration for real-time implementations of TSM and PSM algorithms. The purpose of this paper is to illuminate some of the problems which arise in a real-time context as well as to provide novel solutions to these issues. Here, a real-time software based framework is presented, which allows independent pitch and time stretching with almost unperceivable latency. The approach is based on a modified phase vocoder with optional phase locking and an integrated transient detector which enables high quality transient preservation in real-time. The paper is structured as follows: section 2 outlines basic TSM and phase vocoder theory; section 3 looks briefly at peak locking in the phase vocoder; section 4 introduces a tiered buffer scheme to allow real-time audio processing using a 75% overlap; section 5 deals with transitional artefacts associated with changing parameters in real-time; section 6 introduces a computationally efficient novel pitch shifting method; section 7 describes a novel method for real-time inline transient preservation. A brief evaluation and conclusions follow.

2. PHASE VOCODER FUNDAMENTALS

The phase vocoder was first introduced in [1] and a comprehensive tutorial outlining the theory is presented in [2]. The phase vocoder is a frequency domain technique which can be used to carry out time scale modification of audio. The Fourier transform interpretation of the phase vocoder is mathematically equivalent to a short time Fourier transform (STFT) [3] which segments the analysed signal into overlapping frames which are separated by a certain ‘hop size’. Within phase vocoder implementations, TSM is achieved by varying the analysis hop size (*Ra*) with respect to the resynthesis hopsize (*Rs*) such that the time scaling factor is calculated as: $\alpha = Rs/Ra$.

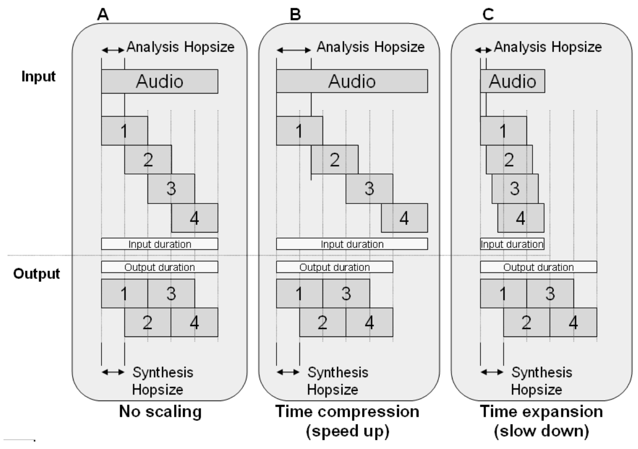

Figure 1: *Three audio segments of various lengths are time scaled to the same output duration. 'A' depicts no time scaling. ‘B’ illustrates time scale compression and ‘C’ illustrates time scale expansion*

From Figure 1 it can be seen that if *Ra* is set equal to *Rs*, no time scaling is achieved, whilst $Ra < Rs$ will result in timescale expansion (slow down), and $Ra > Rs$ will result in timescale compression (speed up). Although, keeping either *Ra* or *Rs* fixed is feasible, it is recommended that *Rs* is fixed and *Ra* is varied in order to avoid amplitude modulation at the output.

In the context of the phase vocoder, a Fast Fourier Transform (FFT) is used to obtain a complex frequency domain representation of the signal. In equation (1), the transform is given as follows:

$$X(t_a^u, \Omega_k) = \sum_{n=-\infty}^{\infty} h(n) x(t_a^u + n) e^{-j\Omega_k n} \qquad (1)$$

Where *x* is the original signal, *h(n)* is a windowing function (typically a Hanning), and $\Omega_k = 2\pi k / N$ is the centre frequency of the kth vocoder channel (bin) in radians per sample, where *N* is size of the FFT. Equation (1) is evaluated for $0 \le k < N$. Also, $t_a^u = uRa$, where *u* is the frame index and *Ra* is the analysis hop size. In order to achieve acceptable TSM output quality, a frame size *N* must be chosen such that an adequate resolution is present to resolve the lowest frequency components expected in the signal. For example, at a sample rate of 44.1 KHz, a frame of length 4096 will yield a resolution of approximately 10.71 Hz per vocoder channel (bin). In practice, a smaller frame size is not sufficient for full bandwidth music but may be sufficient for speech. Given that a windowing function, *h(n),* is applied to reduce spectral leakage during the analysis phase, the synthesis frames must be overlapped if a useable resynthesis is required. Although a 50% overlap ($N/2$) with a Hanning window would yield a perfect resynthesis if no time scaling were applied, it is not generally sufficient for TSM.

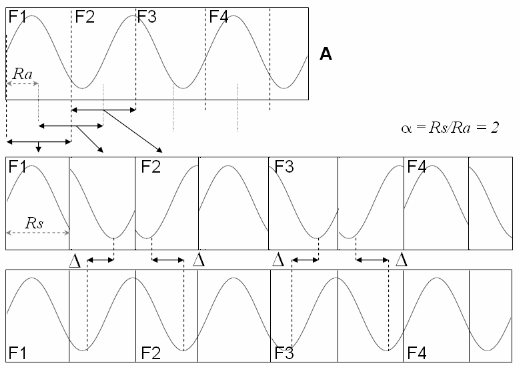

Figure 2: *Simplified illustration of time scale expansion for $\alpha = 2$. 'A' represents the original signal. 'B' illustrates how the analysis frames are remapped to the time scaled output. The discontinuities are present since the frame phases have not been updated. 'C' illustrates the synthesised time scaled signal with phase updates included. The Δ represents the necessary phase shifts required.*

Effectively within the phase vocoder, analysis frames are *'remapped'* along the time axis resulting in newly constructed synthesis frames, each with a modified phase spectrum, in order to ensure that the synthesis frames maintain continuality or 'phase coherence' through time. The fact that the phase spectrum of each frame must be modified implies that the windowing function will also be affected. For this reason, a resynthesis window is necessary and a 75% overlap ($3N/4$) is recommended to avoid modulation at the output. This will result in the output having a constant gain factor of approximately 1.5 which can easily be compensated by multiplying all samples by the reciprocal of 1.5. An overlap of 75% corresponds to a hop size of $N/4$ samples. It is important to note that this corresponds to the fixed synthesis hop size, *Rs. Ra* will only be equal when α=1. For reasons discussed previously, the analysis hopsize will be varied as a function of α in order to achieve time scale modification. It should be appreciated then, that a precomputed short-time Fourier Transform (STFT) will not suffice for implementations where TSM is achieved by varying *Ra* with respect to *Rs.*

In order for the synthesis frames to overlap synchronously, the frame phases must be updated such that phase continuity is maintained between adjacent output frames. Figure 2 illustrates a simplified time expansion scenario; note that the example in Figure 2 does not contain overlapping synthesis frames. As can be seen in the above figure, time scale expansion is achieved by analysing the signal at intervals *Ra,* where *Ra* < *Rs.* Regardless of the value of *Ra, Rs* remains fixed at $N/4$. As can also be seen from Figure 2 B, remapping the frames alone is not sufficient to achieve TSM. Suitable synthesis phases for each frequency bin in each frame must be calculated such that a phase coherent output is generated as in Figure 2 C.

The standard method used to calculate suitable synthesis phases involves calculation of the instantaneous frequency of each bin in radians per sample. Having obtained the instantaneous frequency, it is possible to predict the expected phase of any component for a given synthesis hop size. Given that the frequency content of both music and speech is stationary only over short periods, phase estimates will decrease in accuracy as the hop sizes increase.

The most accurate way to achieve this is by firstly calculating the principal argument of the heterodyned phase increment between adjacent analysis frames as defined in equation 2.

$$\Delta_p \Phi_k^u = \angle X(t_a^u, \Omega_k) - \angle X(t_a^{u-1}, \Omega_k) - R_a \Omega_k \qquad (2)$$

where $\angle X(t_a^u, \Omega_k)$ and $\angle X(t_a^{u-1}, \Omega_k)$ represent the phases of the current and previous analysis frames respectively. So, equation (2) effectively defines the deviation between the observed phase differences between adjacent analysis frames, and the predicted phase differences for the bin centre frequencies (Ω_k) over the hop period *Ra.* $\Delta_p \Phi_k^u$ is then the principle argument of the heterodyned phase increment such that it is in the range $-\pi$ to π. The instantaneous frequency in radians per sample is then given by equation (3).

$$\hat{\omega}_k(t_a^u) = \Omega_k + \frac{1}{Ra} \Delta_p \Phi_k^u \qquad (3)$$

where, $\Omega_k = 2\pi k / N$, is the centre frequency of the kth bin and $\hat{\omega}_k(t_a^u)$ represents the instantaneous frequency in radians per sample of each bin within the analysis frame. In order to calculate the phase spectrum for the synthesis frame at the time scaled output, we simply multiply the instantaneous frequency by the synthesis hop size *Rs,* and add the result to the synthesis phases from the previous frame as in equation (4). This is known as phase propagation or phase updating and is defined as follows:

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^{u-1}, \Omega_k) + R_s \hat{\omega}_k(t_a^u) \quad (4)$$

In order to resythesise the output frame, the magnitude spectrum obtained from the current analysis frame, $\left|X(t_a^u, \Omega_k)\right|$, and the updated phases, $\angle Y(t_s^u, \Omega_k)$ are used as in equation (5). Upon resynthesis using an iFFT, each frame is re-windowed using *h(n)* and overlapped with the previous frames by a factor of $3N/4$ samples (75% overlap).

$$y(t) = h(n) \sum_{k=-\infty}^{\infty} \left|X(t_a^u, \Omega_k)\right| \cdot e^{j\angle Y(t_s^u, \Omega_k)} \quad (5)$$

Equations (1) through (5) represent the fundamental operation of the phase vocoder. Although, the time scaled output is horizontally phase coherent, the timbral quality is often described as sounding *'phasey'*, '*washy*' or '*distant'* and is generally not regarded as natural sounding. Particularly noticeable, is how transients are affected by the phase vocoder. Sharp attacks from melodic and percussive instruments become smeared to an unacceptable degree where all dynamics have been lost. In speech, plosives and stop consonants become softened and slurred, sounding as if the speaker was drunk. The above mentioned artifacts can be directly attributed to the fact that standard phase vocoder only attempts to achieve an optimal phase relationship between adjacent frames; this is termed horizontal phase coherence. However, the pursuit of horizontal phase coherence has a profoundly negative affect on vertical phase coherence which is the relationship between the phases of frequency components within a single frame. Maintaining vertical phase coherence is an important consideration in order to achieve natural sounding TSM. The next section outlines a well known method which addresses the vertical phase coherence issue.

3. PEAK LOCKING

When a single sinusoidal component is analysed using a 'windowed' FFT, generally, more than one frequency bin is excited. This is attributed to spectral spreading. A close relationship exists between the phases of the frequency bins that are excited during analysis; in [4] it is noted, for example, that when a symmetrical window is applied to a sinusoid, adjacent frequency bins exhibit phase differences of +/- π (when the FFT length matches the analysis frame length). For more complex signals, a phase relationship still exists between bins of a single FFT frame, but the relationship can be difficult to predict.

Traditional phase vocoder theory treats all frequency bins as 'sinusoidal' components. During time-scale modification the phases of all bins are, from this perspective, updated by maintaining horizontal phase coherence between corresponding frequency bins between successive synthesis frames. This approach neglects the important phase relationship that exists between frequency bins within the frame itself resulting in what is referred to as a loss of vertical phase coherence in the synthesis frames. As a consequence, a reverberant or '*phasey*' artefact is introduced into the time-scaled signal. The improved phase vocoder [4] explicitly attempts to identify sinusoidal frequency bins in FFT frames by a simple peak picking process within the magnitude spectrum. The phases of these truly sinusoidal peak frequency bins are then updated in the traditional manner, i.e., by maintaining horizontal phase coherence between corresponding peak frequency bins of successive frames. The non-sinusoidal frequency bins are then updated by maintaining the phase difference that existed between each bin and its closest peak/sinusoidal frequency bin. So, if Ω_{kP} represents the centre frequency of the closest dominant peak to a non-sinusoidal (non-peak) bin of frequency Ω_k, the non sinusoidal component is given a phase such that it's relationship to the peak bin is the same in the synthesis frame as it was observed in the analysis frame. This is known as peak locking and equation (6) describes its action.

$$\angle Y(t_s^u, \Omega_k) = \angle Y(t_s^u, \Omega_{kP}) - \angle X(t_a^u, \Omega_{kP}) - \angle X(t_a^u, \Omega_k) \quad (6)$$

4. BUFFER SCHEMES

Until this point, the considerations for offline and real-time implementations of the phase vocoder have been largely similar. One of the key issues in a real-time implementation of TSM is the choice of buffer scheme. In offline processing, the entire signal is overlapped and concatenated before playback. Since the buffer holding the output signal is non volatile, newly processed frames can be easily overlapped with the samples from previous iterations. However, in a real-time environment, a constant stream of processed audio must be outputted and consecutive output frames must be continuous. In order for seamless concatenation, the boundaries of each output frame must be at the constant gain associated with the overlap factor in order to avoid modulation. The method presented below addresses this concern. For reasons discussed in previous sections, a 75% overlap is recommended. This effectively means that at any one time instant, 4 analysis frames are actively contributing to the current output frame. This might seem to imply that we need to process and overlap 4 frames of length *N* before we can output 1 frame but this is not necessarily so.

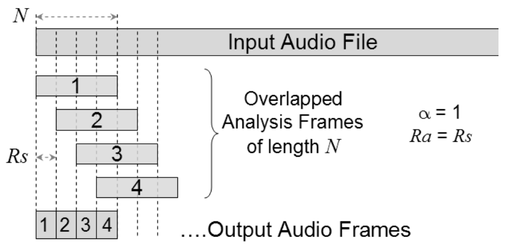

Figure 3: *The relationship between input and output frames for α=1*

In Figure 3, the audio to be processed is divided up into overlapping frames of length *N.* In order to output a processed frame of length *N,* 4 full frames *would* need to be processed and overlapped. This leads to considerable latency from the time a parameter change is affected to the time when its affects are audible at the output. However, given that the synthesis hop size is fixed at *Rs* equal to $N/4$, we can in fact load and process a single frame of length *N*, output ¼ of it, and retain the rest in a buffer to overlap with audio in successive output frames. The output buffer scheme required to achieve this is as follows. Firstly a buffer of length *N* is required in which the current processed frame (with synthesis window applied) is placed. Three additional buffers of length $3N/4$, $N/2$ and $N/4$ will also be required to store remaining segments from the 3 previously processed frames. Each output frame of length $N/4$ is then generated by summing samples from each of the 4 buffers described above. Figure 4, shows how the buffer scheme works.

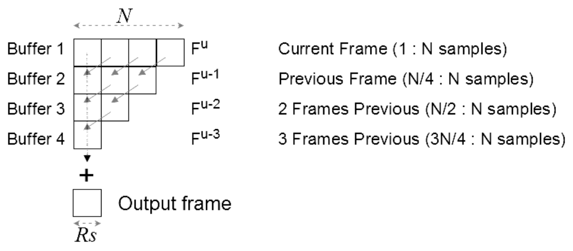
N
Buffer 1
Buffer 2
Buffer 3
Buffer 4
F^u
F^u-1
F^u-2
F^u-3
Current Frame (1 : N samples)
Previous Frame (N/4 : N samples)
2 Frames Previous (N/2 : N samples)
3 Frames Previous (3N/4 : N samples)
+
Output frame
Rs

Figure 4: *Real-time output buffer scheme using a 75% overlap. The gray arrows indicate how each segment of each buffer is shifted after the output frame has been generated.*

Referring to Figure 4; on each iteration, a full frame of length *N* is processed and placed in buffer 1. Remaining samples from the 3 previous frames occupy buffers 2, 3 and 4. The required output frame, S^u, of length $N/4$ is generated as defined in equation (7).

$$S^u(n) = F^u(n) + F^{u-1}(n+\frac{N}{4}) + F^{u-2}(n+\frac{N}{2}) + F^{u-3}(n+\frac{3N}{4}) \quad (7)$$

$$\forall n \qquad 1 \le n \le \frac{N}{4}$$

From equation (7) it can be seen that the output frame is generated by summing the first $N/4$ samples form each buffer. Specifically, buffer 2 contains the remaining $3N/4$ samples from the previous frame (F^{u-1}). Buffer 3 contains the remaining $N/2$ samples from 2 frames previous (F^{u-2}), and buffer 4 contains the remaining $N/4$ samples from 3 frames previous (F^{u-3}). Once the output frame has been generated and outputted, the first $N/4$ samples in each buffer can be discarded. Now, the data in all buffers must be shifted in order to prepare for the next iteration. The gray arrows in Figure 4 illustrate how each segment of each buffer is shifted in order to accommodate a newly processed frame in the next iteration. The order in which the buffers are shifted is vital. Firstly, buffer 4 is filled with the remaining $N/4$ samples from buffer 3; buffer 3 is filled with the remaining $N/2$ samples from buffer 2, and finally, buffer 2 is filled with the remaining $3N/4$ samples from buffer 1. Buffer 1 is now empty and ready to receive the next processed frame of length *N*. The result of this scheme, is that ¼ of a processed frame will be outputted at time intervals of *Rs* which is equal to $N/4$ samples.

Using the suggested frame size of 4096 samples, the output will be updated every 1024 samples which is approximately equal to 23.2 milliseconds. The application presented later allows for near real-time parameter changing of both the timescale modification factor and the pitch shifting factor independently. It is the case that the audio will be processed with newly updated parameters every 23.2 milliseconds but this does not infer that the audible latency will be the same. The latency will most certainly be larger than this and depends on the time required to access and write to hardware buffers in the audio interface. In general however, it is possible to achieve latencies < 40-50ms.

5. PARAMETER CHANGING AND TRANSITIONS

When a fixed time-scale factor is being applied to an entire audio signal both the R_a and R_s parameters remain fixed. The analysis and synthesis time-instants, t_a^u and t_s^u (i.e. the position in time of any analysis or synthesis frames can be defined as uR_a and uR_s, respectively, where *u* is an incrementing integer representing a sequence of frames as in equation (1). For real-time implementations, where the time-scale factor, *α*, may be varying, this definition will introduce distortions into the time-scaled output. One reason for this is because analysis frames will be incorrectly mapped to their corresponding synthesis locations due to a warping of the synthesis time line as *α* is varied. The solution is to redefine $t_a^u = uRa$ as $t_a^u = t_a^{u-1} + \alpha Rs$. This ensures that the current analysis frame position t_a^u, is always updated correctly. Effectively, the position in time of the current analysis frame is always related to both the previous analysis frame and the current time scaling factor, *α*.

Although it is favourable, to vary the analysis hop, R_a, and fix the synthesis hop, R_s, to achieve TSM, it can result in inaccurate frequency estimation for time-scaling factors, $\alpha < 1$. This is the case where, the output is being time scale compressed, i.e., the signal is being speeded up, and as such the analysis hop size *Ra* > $N/4$. Once the distance between analysis frames exceeds $N/4$, it becomes impossible to accurately predict the amount of phase unwrapping to be applied during the frequency estimation stage of the horizontal phase update procedure in equation (2), therefore resulting in inaccurate synthesis phase estimates. In addition to this, when *α* is varied over time, the accuracy of the instantaneous frequency estimates (equation 2) also varies. This leads to momentary artefacts whenever the time scale factor, *α*, is changed. Effectively, the transitions between frames with different TSM factors are not perceptually smooth despite the windowed overlapping scheme. The solution to both of these problems is to ensure that the instantaneous frequency estimates are always derived using the phase differences between the current analysis frame and a frame one synthesis hop back from the position of the current analysis frame, $\angle X\left(t_a^u - R_s, \Omega_k\right)$. It should be noted that this is not the same as the previous analysis frame. Although, an extra FFT and an extra buffer is required to obtain the phases of this frame, it guarantees that phase unwrapping errors will not be present and that the instantaneous frequency estimates will be consistent regardless of variation in *α*. The phase update equation (equation 4) is now redefined in (8).

$$\angle Y\left(t_s^u,\Omega_k\right)=\angle Y\left(t_s^{u-1},\Omega_k\right)+\left(\angle X\left(t_a^u,\Omega_k\right)-\angle X\left(t_a^u-R_s,\Omega_k\right)\right) \quad (8)$$

I.e., the synthesis phases of the current frame are equal to the phases of the previous synthesis frame plus the difference between the current analysis frame and a frame one synthesis hop back from the position of the current analysis frame.

For the case where vertical phase coherence is to be maintained, peak locking can be used, and only the sinusoidal or peak frequency bins are updated using equation (8), with all other bins updated as in equation (6). It is important to note that the synthesis magnitudes are still determined as $\left|Y\left(t_s^u,\Omega_k\right)\right|=\left|X\left(t_a^u,\Omega_k\right)\right|$. This method of phase updating actually removes the need to estimate the instantaneous frequency entirely. However, for the case where pitch scale modification is required, using the real-time approach described in the next section, calculation of instantaneous frequency will still be necessary. Nonetheless, the same 'hop-back' method described above is still used to avoid phase unwrapping errors and to maintain smooth pitch and time scale transitions. This will be discussed in the next section.

A similar phase update procedure has previously been proposed in [5] in which time-scale modification is achieved through the insertion and deletion of entire frames. Since the approach proposed here uses a variable analysis hop size, *Ra*, it has the advantage of maintaining better estimates of the magnitude spectrum, thereby greatly reducing the possibility of removing or repeating perceptually salient characteristics within the time-scaled signal.

6. REAL-TIME PITCH SHIFTING

The simplest method to shift the apparent pitch of a signal is by interpolating or decimating the time domain signal. The resulting signal, although pitch shifted, is also shortened or lengthened by the reciprocal of the interpolation/decimation factor which we will call *β*. A common technique used to shift the pitch and maintain the apparent duration or time base, is to first pitch scale the signal using interpolation/decimation, following which, complimentary time scale modification is applied to restore the original length of the signal. This is easily achievable and solutions have been proposed in an offline context [6], but becomes difficult to address in a real-time context. If both pitch shifting and time scale modification is required simultaneously, the problem becomes more difficult since time scaling is required for 2 alternate operations (pitch and time scaling) within the same frame. One general approach [7] is to calculate the required compensatory time scale factor required, such that when the signal is both time scaled and interpolated for any *α* and *β* factors, the resultant signal is both the required pitch and length. This compensatory factor is simply calculated as *α β,* for example, pitching up by a factor of 2 and simultaneously time scale expanding by a factor 2 requires an overall TSM factor of 4. Bear in mind, that in a real-time context these operations must be carried out within a single frame interval (23ms). Two issues arise here; firstly, the computational requirements are directly related to the product of *α* and *β*, since each frame must now be time-scaled internally to compensate for pitch shifting. This makes real-time operation unfeasible for large products of *α β*. Secondly, the length of the resultant frame is no longer fixed as is required for the output buffer scheme presented above. Instead, an additional buffer must be used in order to handle the overflow if the resultant frame exceeds *N* samples. If (*α β* < 1), the resultant frame will be smaller than the required *N* samples. This is known as buffer under run. In this case, more input frames need to be processed until there are sufficient samples to generate an output frame. These issues are not necessarily detrimental to real-time operation but can make the output unpredictable, added to which the solutions are computational intensive.

Here we present a novel method for real-time pitch shifting designed to operate within the TSM framework described above. The computational requirements are not dependent on the *α* and *β* factors and the method guarantees that a fixed frame length can be generated independent of the time and pitch scale factors used. Furthermore, no additional buffers are required, and no inter-frame time scaling is required. We have approached the problem by carrying out the pitch shifting using linear re-sampling in the time domain in real-time, after which the standard phase vocoder theory can be applied using a modified phase update equation which incorporates the pitch scaling factor *β*. In order to generate a pitch shifted frame of known length, we simply interpolate or decimate the input time domain signal over the range t_a^u to $t_a^u + N\beta$. This results in a time domain frame of required length *N* which has been generated by interpolating or decimating $N\beta$ samples by a pitch scaling factor equal to *β*. This immediately resolves 2 of the problematic issues raised above. Note that this is only possible where random access to the input file is permitted. Frame based host SDKs such as VST will not facilitate this. Figure 5 below illustrates this simple procedure.

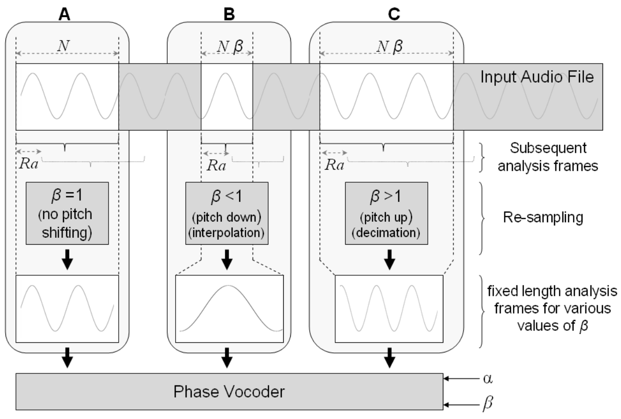
A
B
C
N
N β
N β
Input Audio File
Ra
Ra
Ra
Subsequent analysis frames
β =1 (no pitch shifting)
β <1 (pitch down) (interpolation)
β >1 (pitch up) (decimation)
Re-sampling
fixed length analysis frames for various values of β
Phase Vocoder
α
β

Figure 5: *The real-time resampling method used for obtaining fixed length pitch shifted frames. 'A' illustrates no pitch change, 'B' illustrates pitching down and 'C' illustrates pitching up*.

This interpolated/decimated frame now constitutes an analysis frame which can readily have arbitrary time scaling applied using the modified phase update equations presented below. The goal now is to estimate the phase propagation required to allow successive interpolated frames to be updated such that the pitch shifted and time scaled output is horizontally phase coherent. Recall equation (8), which was introduced as a preferred method to ensure reliable wrapped phase difference estimates. This was achieved by using an extra FFT to estimate the phases of the frame exactly one synthesis hop back from the current analysis frame, thereby allowing the phase differences to be estimated over a known fixed interval equal to *Rs*. Since the *'apparent'* analysis hop is now equal to the synthesis hop, this removes the need to calculate the instantaneous frequency. It is important to note however that the actual value of *Ra* is still variable. However, as stated previously, the method for estimating synthesis phases used in equation (8) does not support pitch shifting. In order to estimate suitable synthesis phases for pitch shifted frames, the instantaneous frequency must be calculated as follows. A method to calculate the heterodyned phase increment for pitch shifted frames is given by equation (9). Note that the interpolation (pitch shifting) factor, *β*, is now included in the equation.

$$\Delta_p\Phi_k^u = \angle X(t_a^u, \Omega_k) - \angle X(t_a^u - Rs, \Omega_k) - \left(\frac{1}{\beta} R_s \Omega_k\right) \quad (9)$$

where $\angle X(t_a^u, \Omega_k)$ and $\angle X(t_a^u - Rs, \Omega_k)$ represent the phases of the current analysis frame and an analysis frame exactly one synthesis hop back from the current value of t_a^u. $\Delta_p\Phi_k^u$ is then the principle argument of the heterodyned phase increment of the pitch shifted frame such that it is in the range $-\pi$ to π. Since the frames have been interpolated or decimated (resulting in frequency shifts) they will no longer exhibit the expected phase derivatives over a given hop, *Rs*. To calculate the correct phase increment, the hop must also be multiplied by the reciprocal of the pitch scaling factor, *β*. The instantaneous frequency in radians per sample of the pitch shifted frame is then given by equation (10).

$$\hat{\omega}_k(t_a^u) = \Omega_k + \frac{\beta\Delta_p\Phi_k^u}{Rs} \quad (10)$$

Again, since interpolation/decimation will cause a frequency shift, the pitch shifting factor, *β*, must be incorporated into the instantaneous frequency equation above. Note that, as oppose to equation 3, we divide the phase deviation, $\beta\Delta_p\Phi_k^u$, by *Rs* instead of *Ra*, because the method used to calculate phase difference in equation (9) uses two frames separated by a fixed distance, *Rs*. The standard phase update method given in equation (4) can now be used. The standard method for peak locking can also be applied as discussed previously. Note that the advantages of using equation (8) for phase updating have already been incorporated in equation (9) above. We now have a set of modified phase vocoder equations which can be used to allow real-time pitch shifting and time stretching simultaneously. The key advantage of using this method for pitch shifting is that compensatory time scaling is not required, instead the pitch scaling factor, *β*, is incorporated in the phase update equations. This guarantees that the computational load remains fixed and predictable for any combination of time and pitch scaling factors. It is important to note that this pitch shifting method does not preserve formants.

7. REAL-TIME TRANSIENT PRESERVATION

The method so far results in a real-time algorithm capable of high quality time and pitch scale modification. We have also addressed the vertical phase coherence issue by introducing peak locking in section 3. Although peak locking does contribute to maintaining the timbral quality of transients during TSM, it gets subjectively worse as *α* is increased. Logically, musical notes from a single instrument can have varying durations and sustains, but percussive instruments such as the snare drum cannot produce sounds of varying duration. As music is slowed in time naturally, the percussive events become spaced by larger intervals associated with the tempo. The phase vocoder does not take this into account; instead, it will compress or expand the duration of each individual percussive event in exactly the same way as all other audio content. Essentially the phases are updated such that horizontal phase coherence is maintained. Figure 6 below illustrates how changing component phases during transient events can lead to transient smearing.

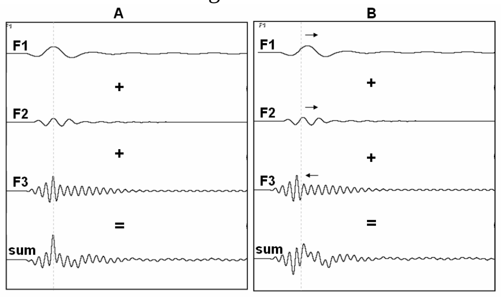

Figure 6: *The effects of time scaling transients*

In the figure 6, 'A' Illustrates three time domain frequency components with no phase alterations. Note how the alignment of peaks sum together to form a sharp transient. Then, 'B' illustrates how altering the phases within the frame can have a negative effect on vertical phase coherence. Note how the transient is smeared when the components are summed during resynthesis. Essentially, transients should not be time-scaled if a natural sounding out put is required. Transient representation and reconstruction has been addressed in the past [8] [9]. In [9], a solution was proposed whereby the signal was separated into steady state regions and transients regions using multi-resolution analysis. The steady state portions only were subjected to time scaling with phase locking (using analysis phases directly as synthesis phases) applied at transient points. The separated transients are then remapped directly into the signal. This results in significantly better output quality for music. The technique was presented as an offline process but something similar can be achieved in real-time.

The approach taken in this paper is to attempt to identify the transients automatically in real-time. Upon detection of a transient, the time scale factor, *α*, is automatically returned to '1' (no scaling), and the analysis phases are mapped directly to the synthesis phases (phase locking) for the duration of the transient. When the transient has passed the time scale factor is automatically reset to the *α* value prior to the transient. Transients represent an ideal place to lock the phases since any discontinuities introduced to the time scaled signal will be masked by the transient itself. Of all transients, drum strikes tend to be affected in the most perceptually objectionable way and so we aim here to preserve drum based transients first and foremost.

With this in mind we require an onset detector which is not concerned with measuring the rapid rises in energy; but rather an onset detector that measures the broadband nature or "percussivity" of the onset, independent of the actual energy present. In this way, drum hits of varying velocity will be detected equally and high energy, band localised events will not be mistaken for percussive onsets. The percussive onset detector described below was first introduced in [10]. In order to identify an analysis frame as a transient, the log difference of each frequency component between consecutive frames is first calculated as in equation (11). This measure effectively tells us how rapidly the spectrogram is fluctuating.

$$X_f(t_a^u,k) = 20\log_{10}\frac{\left|X(t_a^u,k)\right|}{\left|X(t_a^u - Rs,k)\right|} \tag{11}$$

where $X_f(t_a^u,k)$ is the log energy difference between frames separated by Rs, and where t_a^u is the current analysis frame instant. In order to detect the presence of a drum we define a percussive measure given in equation (12).

$$Pe\left(t_a^u\right) = \sum_{k=1}^{N/2}\left[\begin{array}{ll} P\left(t_a^u,k\right) = 1 & if \quad X_f(t_a^u,k) > T_1 \\ P\left(t_a^u,k\right) = 0 & otherwise \end{array}\right] \tag{12}$$

where, T_1 is a threshold which signifies the rise in energy measured in dB which must be detected within a frequency channel before it is deemed to belong to a percussive onset. Effectively equation (12) acts like a counter; $Pe\left(t_a^u\right)$ is simply a count of how many bins are positive going and exceed the threshold, T_1, and $P\left(t_a^u,k\right)$ contains a '1' if the threshold condition is met and zero otherwise. In order for the frame to be declared a transient, $Pe\left(t_a^u\right)$ must exceed a second threshold T_2 as follows: $u^T = u \quad if \; Pe\left(t_a^u\right) > T_2$. In practice we have found that $T_1 = 6dB$ and $T_2 = {}^{3N}\!/_4$ give satisfactory results for most popular music. Note that the actual energy present in the signal is not necessarily the defining factor of a transient. We assign a probability based on a measure of how "broadband" or percussive the onset is. The figure below shows the effectiveness of this approach.

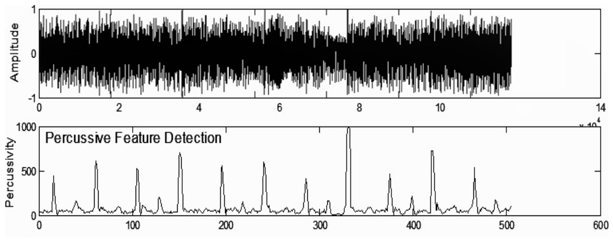

Figure 7: *Here a highly dynamically compressed signal of a rock band is depicted in the top plot. The bottom plot shows the output of the percussive onset detector.*

Note that the percussive feature detection function described can also detect the low amplitude hi-hat strikes between the kick and snare events. Despite the fact that the signal it self has little dynamic range, the percussive feature detector is rarely prone to false detections which makes ideal for transient detection in time scaling. Furthermore, it can easily be implemented within the current framework since the only requirement is that the current and previous frame magnitudes are available.

7.1. Transient Hopping

Upon detecting a transient, the time scale factor, α, is automatically returned to '1', inhibiting TSM momentarily. We term this method 'transient hopping'. In addition the frame phases are locked and the frame is mapped directly to the output. This mechanism preserves the transient and ensures that it is reproduced unaffected at the output. Given that the recommended analysis frame length is 4096, and transient events are significantly shorter, it would seem feasible to cease time scaling for the transient frame only, but this is not strictly true. Given that we are using a 75% overlap, i.e., Rs = 1024, the observed transient will exist in 4 consecutive frames due to overlapping. In order to preserve the transient correctly, the TSM factor, α, must remain at a value of '1' until all overlapping frames have passed the transient, i.e., 4 frames. Equation (13) describes this action.

$$\alpha^T = \begin{cases} 1, & if \;\; u - u^T < 4 \\ \alpha, & otherwise \end{cases} \tag{13}$$

where u is the current frame index and u^T is the index of the last frame containing a transient, and α^T is the modified time scale factor around the transient. Figure 8 below shows a signal and its time scaled counterparts. The phase vocoder with transient preservation (plot 2) keeps the sharp attack of the transient whereas the standard phase vocoder (plot 3) smears the start of the transient resulting in subjectively poor quality resynthesis.

Analysing Figure 8 in a little more detail, it can be seen that when transient preservation is used, the time scaled signal is a little shorter in duration than expected. This is due to the fact that no time scaling has been applied during the transients and so the local time scale factor is reduced slightly.

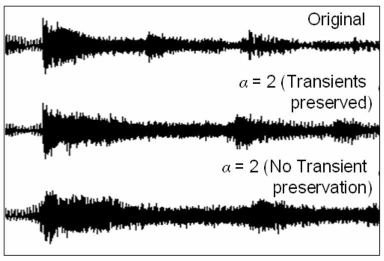

Figure 8: *A signal time scaled by a factor of 2 with and without transient preservation*

This causes a problem. Since the local time scaling factor is always '1' around a transient and α elsewhere, we will have rhythmic fluctuations (due to α fluctuation) depending on the density of transients along the time line. In order to address this issue, we propose a time scale compensation factor which is applied after each transient, such that 4 frames immediately following the transient will be subjected to a time scale factor of (α x 2). This compensates for the loss of 4 time scaled frames during the transient. Equation (13) can then be rewritten as:

$$\alpha^T = \begin{cases} 1, & if \;\; u - u^T < 4 \\ 2\alpha, & if \;\; u - u^T < i \quad \forall i \;\; 4 \le i < 8 \\ \alpha, & otherwise \end{cases} \tag{14}$$

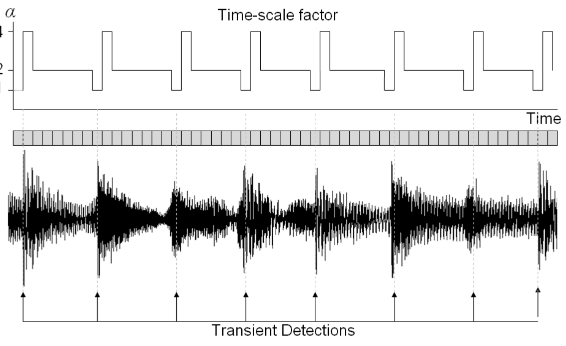

Figure 9: Time scale factor as a function of transient detection using a global TSM factor of α=2

Figure 9 illustrates how the time scaling factor is varied both before and after the transient in order to both preserve the transient and to maintain a constant global time scale factor. The method operates well and achieves a high quality output but can become problematic when multiple transients are located over a short duration. This is observed in drum fills for example; where multiple drum hits are placed in rapid succession. In this case, rhythmic synchronisation can be lost momentarily.

8. CONCLUSIONS

A review of phase vocoder theory with a focus on real-time implementation has been presented. A modified phase vocoder framework has been presented, which allows simultaneous pitch and time scale modification in a real-time context. The key advantage of using this method for pitch shifting is that compensatory time scaling is not required, instead the pitch scaling factor, β, is incorporated in the phase update equations. This guarantees that the computational load remains fixed and predictable for any combination of time and pitch scaling factors. Furthermore, the method for calculating phase differences (equation 9) ensures perceptually smooth transitions when either α or β are varied in real-time. In addition a scheme for real-time transient preservation was introduced which results in a superior output quality to using peak locking alone. However, the real-time transient hopping approach can lead to some unpredictable results in situations where there is a high density of transients spaced closely together in time. The use of variable window sizes at transients could provide superior results. Some video examples of a real-time implementation and application based on the above theory can be viewed at:

http://www.audioresearchgroup.com/main.php?page=Demos

9. ACKNOWLEDGEMENTS

This work was in part supported by the European Community under the Information Society Technologies (IST) programme of the 6th FP for RTD - project EASAIER contract IST-033902. The work was also supported by Enterprise Ireland project IMAAS, CFTD/06/220.

10. REFERENCES

[1] Flanagan J.L. and Golden R.M., “Phase Vocoder”, *Bell System Technical Journal* 45: 1493-1509, 1966.

[2] Dolson, M., “The phase vocoder: A tutorial”, *Computer Music Journal*, vol. 10, pp. 14-27, 1986.

[3] Portnoff, M, “Implementation of the digital phase vocoder using the fast Fourier transform” in *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Volume: 24, Issue: 3On page(s): 243- 248, Jun 1976

[4] Laroche, J.; “Dolson, M., “Improved phase vocoder”, *In Proc. IEEE Transactions on Speech and AudioPprocessing*, vol. 7(3), pp. 323 –332, May 1999.

[5] Bonada, J., “Automatic technique in frequency domain for near-lossless time-scale modification of audio”, 'Proceedings of International Computer Music Conference, Berlin, Germany 2000

[6] McAulay, R. J. and Quatieri, T. F. “Speech Transformations Based on a Sinusoidal Representation”. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-34:6, pp. 1449–1464, August 1986

[7] Laroche, J. "Autocorrelation method for high quality time/pitch scaling". *IEEE Proc. Workshop Appl. of Signal Processing to Audio and Acoustics*, Mohonk, NY., 1993.

[8] Tony S. Verma and Teresa H. Y. Meng, "An analysis /synthesis tool for transient signals," in *Proc. 16th International Congress on Acoustics/135th Meeting of the Acoustical Society of America*, June 1998, vol. 1, pp. 77 -- 78

[9] Duxbury, C., M. Davies, and M. Sandler. “Improved time-scaling of musical audio using phase locking at transients”. *In proc 112th AES Convention*. Convention Paper5530, 2002

[10] Barry, Dan; FitzGerald, Derry ; Coyle, Eugene, “Drum Source Separation using Percussive Feature Detection and Spectral Modulation”, *In Proc. IEE Irish Signals and Systems Conference*, Dublin City University, Dublin, Ireland., 2005