THE SLIDING PHASE VOCODER

Russell Bradford and Richard Dobson and John ffitch
Media Technology Research Centre
University of Bath, UK

ABSTRACT

The Sliding Discrete Fourier Transform (Sliding DFT) is used as the engine of a phase vocoder, to create a Sliding Phase Vocoder (SPV). With a little care this allows very accurate pitch shifting and low latency, and opens a number of possible extensions. We also consider the use of vector parallel processing to make these techniques a viable option.

1. BACKGROUND

In his AES lecture, Moorer[11] considered what might be the state of audio processing in 2020, given the ongoing increase in computer power as defined by Moore's Law; indeed, with the increasing availability of multiple-core devices at consumer prices, it seems likely that if anything, the trend defined by Moore will accelerate rather than decline. His predictions included the routine use of frequency-domain processing of audio where he suggested that the Sliding DFT (SDFT), in which the analysis frame is updated every sample, might offer some advantages. In [3] we described a working implementation of the SDFT, with some initial qualitative assessments that indicated that the SDFT merited further investigation. Accordingly, we only give a brief description in section 2. In this paper we consider the properties of the SDFT and its application to the conventional phase vocoder (pvoc).

The SDFT is computationally expensive despite its advantages, but multiple processor cores on one chip changes the rules; we are also working with ClearSpeed Technology plc[12] on deploying their vector processor hardware in the general area of High-Performance Audio Computing (HiPAC), and the SDFT is a clear target for this work.

2. THE SLIDING PHASE VOCODER

In [3] we presented the basic form of the Sliding Discrete Fourier Transform, with the improved reconstruction. For completeness we repeat the fundamental result here. The idea is to construct the DFT for a window from the DFT for the previous window, one sample earlier. If F_t is the DFT at time t of the discrete signal f_t and the n^{th} bin is denoted by $F_t(n)$, then for a window of size N samples we can calculate the next DFT for the frame one sample later from the current DFT by

$$F_{t+1}(n) = (F_t(n) - f_t + f_{t+N})\, e^{2\pi i \frac{n}{N}}$$

	PVOC	**SPV**
Decimation	250	1
Analysis Frame rate	192	48000
FFT bin bandwidth	± 48	± 24000
Bin Centre Freq	$48m$	$\pm 2\pi(SR/2)$
	$m = 0 \ldots N/2$	

Table 1. Pvoc/SPV comparison: FFT frame size 1000, Sample rate 48000

Note that this requires only N complex multiplications, and is totally parallel over the bins. Also note that N is not constrained to be a power of 2.

The SDFT itself is reasonably efficient, especially considering that the analysis rate is now equal to the sample rate [8]. To serve any musical purpose, some form of processing must be added to the bare SDFT algorithm and this processing will run at the sample rate, and will add considerably to the processing load. Historically, the SDFT has been used primarily for so-called 'zoom-FFT' processing, in which only a few analysis bins covering a narrow frequency band are computed at each update. For audio purposes this has to be expanded to full-bandwidth processing, with the consequent processor load. However the SDFT algorithm is highly suited to parallel computation, the SDFT update calculation being a clear example of a SIMD operation, where the same calculation is applied to each analysis bin, with no (or minimal) data dependencies. Given suitable hardware the SDFT update can be performed in the time of two adds and a complex multiply. Without such hardware, our current investigations are essentially 'proof-of-concept', to test the robustness of the algorithm, and the potential for musically useful processing.

The phase vocoder has a long and venerable history, and remains a highly important process, available in both academic and commercial forms [2, 6]. Its operation and properties are well-known [9, 5], and require no detailed description here. We are however most interested in the similarities and differences with the SPV. Table 1 presents a summary of the primary parameters for the two methods. The single most significant difference introduced by the SPV is the bin bandwidth, which is determined by the analysis rate. In pvoc this is very low, and the bandwidth of a single bin is commensurately narrow. This is commonly described in terms of phase wrapping — the accumulating change of phase in a bin between successive

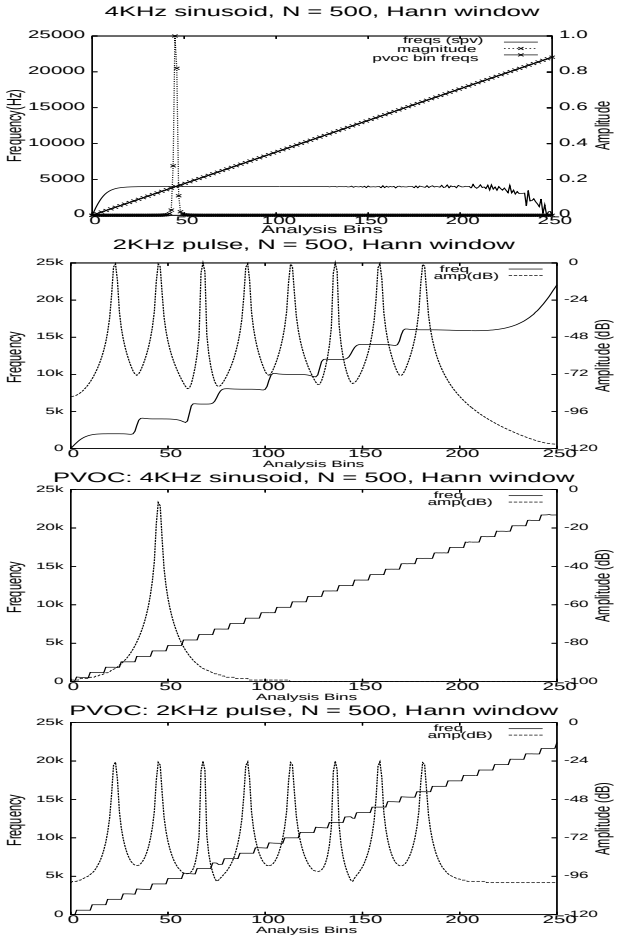

Figure 1. Frequency Profile Example: (a) 4KHz sine, (b) 2KHz pulse, (c) and (d) pvoc equivalents

frames is wrapped round the bin centre frequency modulo 2π. Interpreted as a filter bank [9], each bin corresponds to an elementary bandpass filter, such that there is significant overlap between bins. Where the input contains a prominent frequency component, a small group of adjacent bins will contain frequencies very close together, at or close to the component frequency. We term this phenomenon 'bunching'. In a typical pvoc analysis frame, frequency bunching will typically involve up to 8 adjacent bins for a single isolated source component.

With the SPV, the possible phase change between frames covers the full audio bandwidth, just as an oscillator can span $\pm$Nyquist. The most significant consequence of the narrow bin bandwidth in pvoc is that when bin frequencies are modified (for example, in pitch shifting), the new frequency must be correctly located in a bin that can legitimately contain it. This requirement is obviated in the SPV, since any bin can contain any positive or negative frequency available at the given sample rate. Furthermore, the bunching behaviour observed in pvoc can, in the SPV, involve all bins in a frame. The frequency value calculated for a given bin depends entirely on the input, such that for all practical purposes the theoretical centre frequency of the bin ceases to have any relevance. In the conventional pvoc, the filters are linearly distributed over the frequency range. The difference with the SPV is that the bandwidth of each filter encompasses the whole audio range, so that rather than being arranged side to side, they are in effect stacked vertically.

3. ANALYSIS WINDOWS IN THE SPV

It is well-known that for most DFT-type analysis tasks, it is important to window the sample block to taper the data towards zero at the ends of the block. In conventional block-based pvoc windowing is performed in the time domain, the choice of windows depending on the application, presenting the designer with an unavoidable tradeoff between a narrow main lobe, and suppression of side-lobes[7]. For musical applications, both aspects are important in resolving closely-tuned and low-frequency components, and in distinguishing components from noise. Classic windows include the Blackman-Harris, Hamming and Hann, all members of a large family of windows derived from the basic raised-cosine window. The Kaiser window, though much more complex to calculate, offers a parameter whereby the balance between these parameters can be adjusted to taste, and is often the window of choice in conventional PVOC applications.

With the SDFT, this time-domain method is not available (it would involve total recalculation at each sample), and windowing must be implemented as a frequency-domain convolution. Windows based on cosines are easy to express as convolutions as cosines become shifts in the spectral domain [1], and our work so far has been confined to them, with the Hann window proving the most important in the context of the SPV. While characterisations of analysis windows when applied to a single DFT frame naturally focus on bin magnitudes, as described above, we have found that the SPV offers a significant alternative approach in which otherwise subtle differences between windows manifest as quite pronounced differences in the frequency profile of an SPV frame. In turn, these differences prove to be highly relevant to the frequency-oriented transformations (such as pitch shifting) that may be implemented in the SPV, in which many of the complexities associated with conventional pvoc analysis fall away.

Figure 1 illustrates a typical frequency profile for (a) a single 4Khz sinusoid (SR = 44100) and (b) a band-limited 2KHz pulse-wave (the corresponding amplitude spectrum is superimposed in the plots). By way of comparison, (c) and (d) demonstrate the same signals applied to the conventional pvoc. The latter exhibit the stepwise bunching that is unavoidable in pvoc for the low analysis rates commonly used. Figure 2 presents a comparison between various windows for the pulse-wave source, in the SPV. The special property of the Hann window is easily observed. Most significantly, not only is the bunching associated with a source partial observed as region of zero-slope in the frequency profile, but there are no 'false' zero-slope

[1] The Kaiser window stands somewhat apart from these, being based on Bessel functions rather than on cosines

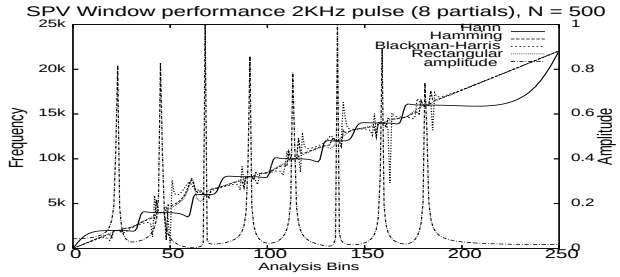

Figure 2. Window comparisons with 2KHz pulse: (a) rect, (b) Hamming, (c) Hann, (d) Blackman-Harris

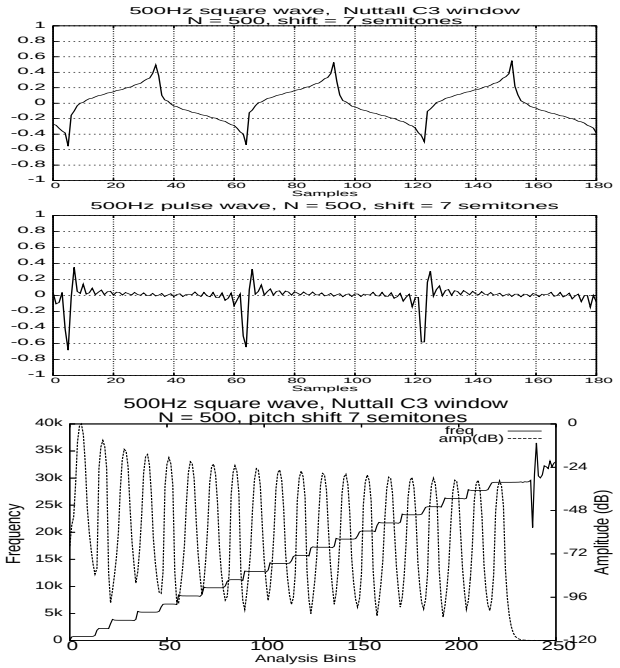

Figure 3. Pitch-shift: (a) b/l square-wave, (b) b/l pulse (c) Nuttall C3 window

regions where there is no corresponding source component. Some other windows have proved to be very close in behaviour, while most generate significant arbitrary changes of slope between partials. We regard the performance of the Hann window as highly significant, as it offers the possibility that transformations that require detection of frequency components may be based solely on the existence or otherwise of zero-slope frequency values, without reference to the presence of peaks in the amplitude profile. While we find that the standard 2nd-order ($\cos^2$) windows work very well, we have tested some higher-order ($\cos^3$) windows such as Nuttall C3 and Blackman-Harris, with the Nuttall window so far exhibiting the best frequency characteristics. That is to say, the frequency profile shows an almost ideal stepwise form with minimal overshoot, evidently corresponding to the substantial suppression of sidebands in the DFT profile itself.

Given this very 'clean' behaviour of the Hann window in the SPV, and also given that SPV resynthesis is by definition that of an oscillator bank, no bin-based corrections are required for the canonical pitch-shift task. It is sufficient merely to scale the bin frequencies as required. Where the frame size is sufficient to fully capture a component partial, phase-shift errors, while still present, are appreciably reduced compared to standard pvoc. Figure 3 presents typical waveforms of pitch-shifted pulse and square wave. Example (c) illustrates the performance of the Nuttall C3 window prior to clipping bins above Nyquist for resynthesis (preserving the source sample rate), exhibiting an almost ideal frequency profile, with virtually no overshoot in the frequency steps, corresponding to almost full suppression of sidebands between the component peaks of the source. Typically, for a given level of phase shift distortion, the SPV can be used with smaller values of N compared to standard pvoc (though we note that as the algorithms are so different, the phase distortions also differ markedly in nature, so the comparison is not a robust one). Since all such modifications are performed at the sample rate, streaming audio-rate frequency-modulation of the input is possible, with results (at least for sinusoids) comparable to those of classic FM synthesis.

4. LATENCY

A well-known obstacle to the general real-time use of the standard phase vocoder is the latency delay incurred through the use of overlapped sample blocks. The primary latency is of a full analysis frame of N samples. A further latency is incurred through the use of overlapped frames, which may amount (depending on the implementation) to an additional delay of $N/2$ samples. In consequence, delays in excess of 25msecs are typical for real-time pvoc processes — more so when one takes into account the additional latency imposed by the audio subsystem.

The SPV (and equally the underlying SDFT) reduces the overlap to one sample, giving an immediate reduction in latency from that factor alone. However, since the DFT frame itself is developed and updated sample by sample, and output is generated similarly, the latency is found to be substantially less than N. While the contents of the frame for the first few samples understandably bear little relation to the source, we find that useful output is available when just one third of the frame has been filled. This appears to be an exact figure — tests with N a multiple of 3 confirm that viable output (directly matching the input signal) commences exactly at sample $N/3$. Figure 4 illustrates the case for $N = 1500$, where useful output starts exactly at sample 500, excluding some startup transient behaviour[2]. The SPV thus reduces the latency imposed by standard pvoc, for a given value of N, by some 75%, with clear and valuable advantages for real-time performance.

5. IMPLEMENTATION DETAILS

Our current implementation of the SPV is as a conventional command-line program running on standard hard-

[2] NB: offline (file-based) pvoc implementations conceal this latency by absorbing input until a complete output frame is ready.

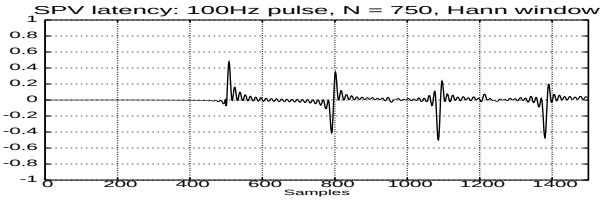

Figure 4. SPV Latency: N-1500

ware, with all calculations and storage using 64-bit floating-point. While the details of the SDFT have been developed from first principles, the phase vocoder component of the SPV (the conversion of the complex output of the SDFT to standard pvoc amplitude and frequency frames) is entirely conventional. We selected Stephan Bernsee's excellent example phase vocoder pitch shifter[1] as the basis for the SPV as it is widely known, and includes few of the distracting complexities of other well-known implementations such as CARL pvoc[4].

However, as noted above, the pitch-shift algorithm can be much simpler in the SPV, requiring no more than scaling of each frequency bin value, and the usual precautions against exceeding the Nyquist limit. This extends to the general matter of frequency-related modifications. We note that in the phase vocoder described by Moore[10], which forms the basis of several public-domain tools, whenever bin frequencies are modified, resynthesis is always performed by means of an oscillator bank so that frequencies may be freely modified over the range. The SPV generalises this approach — the SPV is an oscillator bank as much as it is a phase vocoder.

6. CONCLUSIONS AND FUTURE WORK

We have presented the SPV as the first of a new category of HiPAC processes, targetted at next-generation parallel SIMD hardware, as proposed by Moorer. We do not position the SPV as a tool for use on conventional computing hardware. The SPV is shown to be a fully viable process, demonstrated by the canonical pitch shift task. This proves to be much simpler than with standard pvoc while offering new opportunities for sound transformation. The SPV narrows the distinction between the phase vocoder and the oscillator bank to the degree that the SPV itself *is* an oscillator bank, offering complete freedom of frequency modification to effects developers. The SPV has in turn proved to be also a useful new resource for evaluating analysis windows. The results shown in this paper are preliminary, and much more work is needed to develop this and other aspects further. We also seek to study the remaining phase-related problems familiar to all pvoc users and equally present in the SPV, most particularly the smearing of transients, which remain ineluctably associated with the window size.

In addition to the primary goal of real-time implementation, we consider the SDFT and the SPV offer opportunities to develop a viable constant-Q version, offering the prospect of yet smaller frame sizes, and also improvements with respect to transient smearing. The potential for new classes of frequency-domain processing is exciting. The practicability of frequency-domain FM has been alluded to; we expect many other possibilities to emerge.

7. ACKNOWLEDGEMENTS

This project has been supported by grant funding from the Arts and Humanities Research Council under their "Speculative Research" theme. We are also grateful for encouragement and support from ClearSpeed Technology plc in moving towards real-time implementations of the SPV and other HiPAC processes.

8. REFERENCES

[1] Stephan M. Bernsee. Pitch Shifting Using the Fourier Transform. `http://www.dspdimension.com`, 1999.

[2] Richard Boulanger, editor. *The Csound Book: Tutorials in Software Synthesis and Sound Design*. MIT Press, 2000.

[3] Russell Bradford, Richard Dobson, and John ffitch. Sliding is Smoother than Jumping. In SuviSoft Oy Ltd, Tampere, Finland, editor, *ICMC 2005 free sound*, pages 287–290. Escola Superior de Música de Catalunya, 2005.

[4] Richard Dobson. Developments in Audio File formats. In Ioannis Zannos, editor, *ICMC2000*. ICMA, August 2000.

[5] R. Fischman. The Phase Vocoder: Theory and Practice. *Organised Sound*, 2(2):127–145, 1997.

[6] Electronic Music Foundation. Grm tools. `http://www.grmtools.org/`.

[7] Fredric J. Harris. On the Use of Windows for Harmonic Analysis with Discrete Fourier Transform. *Proceedings of the IEEE*, 66(1), Jan 1978.

[8] E. Jacobsen and R. Lyons. The Sliding DFT. *IEE Signal Processing Magazine*, March 2003.

[9] Jean Laroche and Mark Dolson. New phase vocoder techniques for pitch-shifting, harmonizing and other exotic effects. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, New Paltz, NY., 1999.

[10] F. R. Moore. *Elements of Computer Music*. Prentice Hall, 1990.

[11] James A. Moorer. Audio in the New Millennium. *J. Audio Eng. Soc.*, 48(5):490–498, May 2000.

[12] ClearSpeed Technology plc. ClearSpeed Whitepaper: CSX Processor Architecture. `http://www.clearspeed.com`, Feb 2007.