PD SPECTRAL TOOLKIT

Cooper Baker
Department of Music
University of California San Diego
9500 Gilman Dr. MC 0099
La Jolla, CA 92093-0099

Tom Erbe
Department of Music
University of California San Diego
9500 Gilman Dr. MC 0099
La Jolla, CA 92093-0099

ABSTRACT

The Pd Spectral Toolkit, a set of Pure Data objects for spectral signal processing in OS X, is described and illustrated with example patches. These objects facilitate spectral signal processing by providing commonly used data manipulations, conversions, and utilities that users must implement from scratch when working with spectral data. Additional objects are provided to achieve novel methods of spectral data manipulation, allowing more advanced spectral algorithms to be expressed while requiring fewer objects within a Pure Data patch. A comprehensive list of objects is included, and a web address to download the toolkit, its source code, and an IDE project containing the objects is provided.

1. DESCRIPTION

The Pd Spectral Toolkit is a set of 80 objects for the Pure Data real-time music and multimedia environment (Pd) [13]. The toolkit was developed to facilitate spectral signal processing within Pd by providing a set of objects that perform frequently used data transformations and calculations. While some of these transformations and calculations may be implemented with standard Pd objects, doing so requires users to spend considerable time constructing lower-level boilerplate algorithms by combining dozens of objects. Consequently, composers and audio engineers using Pd to perform spectral data manipulation are encumbered with implementation, and can quickly get lost constructing complex patches that perform relatively simple spectral transformations.

The Spectral Toolkit provides access to a larger number of data manipulation techniques than currently available with standard Pd objects, and affords more concise expression of spectral algorithms within the Pd patching language. By incorporating many lower-level boilerplate operations into individual objects, composers and engineers may spend less time managing lower-level data bookkeeping, and more time implementing complex spectral algorithms, resulting in easier access to advanced spectral techniques and simpler patches. In addition to spectral signal processing objects, the toolkit also contains a few abstractions for spectral data visualization, and a set of utility objects that are useful in non-spectral contexts.

These objects are pedagogically useful and serve to help teachers demonstrate spectral data processing by making algorithms easier to understand, and making implementation and exploration of these algorithms simpler for students. Additionally, the source code of the Pd Spectral Toolkit is available and provides numerous examples of how to use the Pd external object application programming interface, as well as insight into spectral signal processing in the C programming language.

2. OBJECT CATEGORIES

The objects may be broadly categorized into several groups based on their intended uses. The spectral category includes objects that manipulate spectral data and act as building blocks for more complicated spectral algorithms. The conversion category includes objects that translate spectral data into various formats, as well as objects that perform simpler translations between acoustic units. The operator category includes objects that perform complex arithmetic and several other operations. The comparison category includes objects that perform logical comparisons between signal values. Finally, the miscellaneous category includes objects that do not fit anywhere else. The objects in these categories are listed in section 4, and further documentation, including object downloads, help diagrams, source code, and an IDE project may be found online at http://www.cooperbaker.com/pd-spectral-toolkit.

3. EXAMPLES

Several example patches are included with the Pd Spectral Toolkit in order to show its objects at work. Most of these examples comprise familiar spectral algorithms and show how the objects can simplify spectral processing within Pd. Four examples are shown here to demonstrate how the objects can simplify spectral patches.

3.1. Spectral Morph

The first example illustrates a spectral morph algorithm [19] in which the spectra of two input sources are morphed together to create one output spectrum. The *winfft~* object performs a windowed fast fourier transform (FFT) on an input audio signal, creating cartesian coordinate pairs which are then converted to polar coordinate pairs by the *cartopolar~* objects. To achieve morphing, the magnitudes of these pairs are multiplied and the phases are added, resulting in a new set of polar coordinate pairs. The new polar coordinates are converted back to cartesian coordinates by the *polartocar~* object and finally transformed into an audio signal by the *winifft~* object, which performs a normalized inverse windowed FFT.

The *windowing_scheme* subpatch includes the *windower* object, which fills a table with various window functions, and other standard Pd objects to generate messages for the *block~* object. Please see figure 1 for the patch containing these objects.

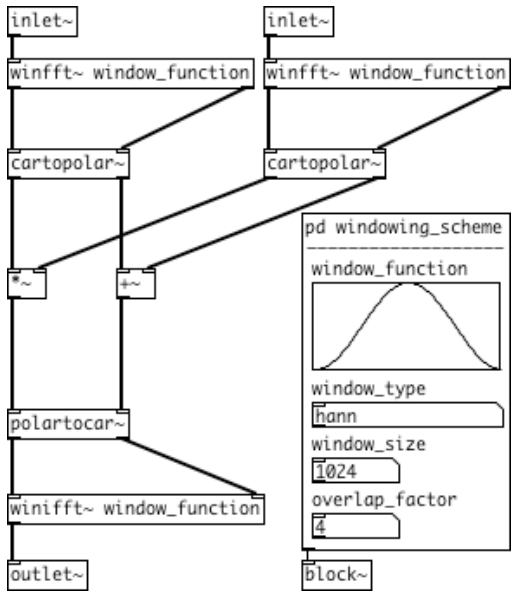

Figure 1: Spectral Morph

3.2. Oscillator Bank Resynthesis and Pitch Shift

Example Two illustrates a simple oscillator bank resynthesis algorithm implemented with objects from the toolkit. The input signal is transformed into cartesian coordinate pairs by the *pafft~* object, which performs a windowed, phase-aligned FFT. Within *pafft~*, phase-alignment is performed by rotating the time-domain signal of each analysis frame prior to transformation, thereby mitigating phase offsets introduced by overlapped FFT analysis. The cartesian pairs are converted to magnitude and instantaneous frequency pairs by *cartofreq~*, then frequency values are scaled by a factor of two to achieve an upward pitch shift of one octave. The magnitude and frequency pairs are interpreted by *oscbank~*, which transforms them back into an audio signal by rendering waveforms from its bank of oscillators. Again, the *windowing_scheme* subpatch contains the *windower* object and function table, as well as objects that configure *block~* and other objects that calculate the overlap factor and control phase-alignment within *pafft~*. Figure 2 shows these objects in a patch.

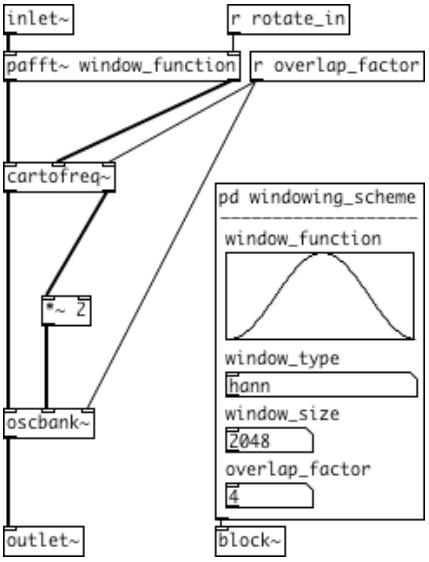

Figure 2: Oscillator Bank Resynthesis

3.3. Classic Phase Vocoder

Example Three illustrates toolkit objects implementing a classic phase vocoder algorithm [4] that time stretches a sample by a factor of four. The *pafft~* performs a windowed, phase-aligned FFT on audio signals emitted from the *play_sample* subpatch, resulting in cartesian coordinate pairs. These pairs are converted to polar coordinate pairs by the *cartopolar~* object, then their phase values undergo several manipulations to create smooth changes between successive spectral frames. First, the *phasedelta~* object computes phase deviations; next, *piwrap~* wraps phases between $-\pi$ and π; third, phase is scaled by a factor of four; finally, *phaseaccum~* accumulates phases to maintain smooth transitions. After these manipulations, the new polar coordinate pairs are converted back to cartesian pairs by *paifft~* which performs a normalized windowed inverse FFT, resulting in a time-stretched output signal. Similar to previous examples, the *windowing_scheme* subpatch generates and contains the window function, controls the *block~* object, and manages signal rotation for *pafft~* and *paifft~*, while also calculating *sample_hop_size* for the *play_sample* subpatch. These objects are shown working together in figure 3.

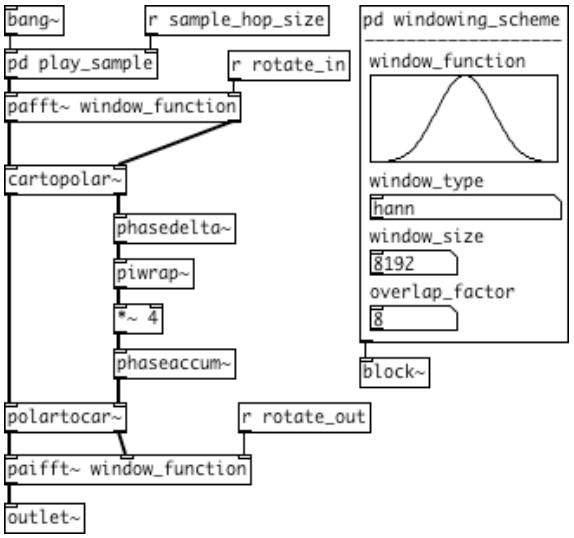

Figure 3: Classic Phase Vocoder

3.4. GEM Waterfall Display

Example Four shows a realtime 3D scrolling waterfall display created with spectral toolkit objects and GEM. Spectral data is stored in an array used by GEM to draw a moving, topological, colorized surface, that represents changing spectral amplitude envelopes. The display may be rotated and zoomed to aid visualization and inspection of spectral data. In addition to the waterfall display, a simpler realtime 2D scrolling colorized sonogram patch is also available for data visualization.

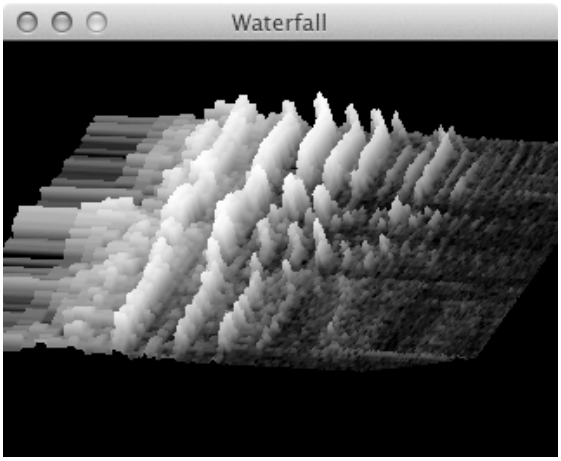

Figure 4: GEM Waterfall Display

4. OBJECT LIST

4.1. Spectral Objects

binindex~ binmax~ binmin~ binmix~ binmonitor~ binsort~ bintrim~ blocksmooth~ freqsieve~ fundfreq~ harmprod~ magtrim~ oscbank~ pafft~ paifft~ partconv~ peaks~ phaseaccum~ phasedelta~ piwrap~ rotate~ valleys~ windower winfft~ winifft~

4.2. Conversion Objects

amptodb~ amptomag~ cartoamp~ cartodb~ cartofreq~ cartomag~ cartophase~ cartopolar~ ctltosig~ dbtoamp~ dbtomag~ degtorad~ degtoturn~ freqtocar~ freqtophase~ freqtopolar~ magtoamp~ magtodb~ phasetofreq~ polartocar~ polartofreq~ radtodeg~ radtoturn~ sigtoctl~ turntodeg~ turntorad~

4.3. Operator Objects

!~ %~ cmplxabs~ cmplxadd~ cmplxdiv~ cmplxmult~ cmplxsqrt~ cmplxsub~ recip~ rounder~ trunc~

4.4. Comparison Objects

!&&~ !=~ !||~ &&~ <=~ <~ ==~ >=~ >~ ||~

4.5. Miscellaneous Objects

bitsafe~ countwrap dspbang~ monitor~ rgbtable scale~ softclip~ tabindex~ terminal

5. FURTHER WORK

These objects provide a solid foundation for building spectral signal processing patches in Pd. However, to make this collection of objects even more robust, objects for spectral peak tracking could be implemented to give users more control over resynthesis. Cepstral techniques are not yet implemented and would provide access to new algorithms for data manipulation. Additional operator and conversion objects would serve to further simplify expression of spectral algorithms in Pd patches, and classic spectral algorithms, such as phase vocoding, could be implemented in their own discrete objects.

6. ACKNOWLEDGEMENTS

Development of the Pd Spectral Toolkit was funded by a Committee on Research grant provided by the University of California San Diego during the 2012 – 2013 school year. The toolkit was created by Cooper Baker under the direction of Tom Erbe. Miller Puckette provided insight into the Pd application programming interface as well as invaluable advice about spectral math and spectral signal processing algorithms.

7. REFERENCES

[1] Bracewell, R., *The Fourier Transform and its Applications, Third Edition*, McGraw Hill, 2000.

[2] De La Cuadra, P., Master, A., Sapp, C., "Efficient Pitch Detection Techniques for Interactive Music", *Proceedings of the International Computer Music Conference*, La Habana, Cuba, 2001.

[3] De Poli, G., Piccialli, A., Roads, C., *Representations of Musical Signals*, The MIT Press, 1991.

[4] Dolson, M., "The Phase Vocoder: A Tutorial", *Computer Music Journal, Vol. 10, No. 4*, 1986.

[5] Flanagan, J. L., Golden, R. M., "Phase Vocoder", *The Bell System Technical Journal*, 1966.

[6] Jaffe, D. A., "Spectrum Analysis Tutorial, Part 2: Properties and Applications of the Discrete Fourier Transform", *Computer Music Journal, Vol. 11, No. 3*, 1987.

[7] Klingbeil, M. *Spectral Analysis, Editing, and Resynthesis: Methods and Applications*, Columbia University, 2009.

[8] Loy, G. *Musimathics: The Mathematical Foundations of Music*, The MIT Press, 2006.

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

[10] Park, T. H., *Introduction to Digital Signal Processing: Computer Musically Speaking*, World Scientific Publishing Co. Pte. Ltd., 2010.

[11] Press, W., Teukolsky, S., Vetterling, W., Flannery, B., *Numerical Recipes: The Art of Scientific Computing, Third Edition*, Cambridge University Press, 2007.

[12] Puckette, M., *Pd Source Code*, http://crca.ucsd.edu/~msp/Software/pd-0.43-3.src.tar.gz, 12/13/12.

[13] Puckette, M., *Pure Data*. http://www.crca.ucsd.edu/~msp/software.html, 01/20/13.

[14] Puckette, M., *The Theory and Technique of Electronic Music*, World Scientific Publishing Co. Pte. Ltd., 2007.

[15] Roads, C., *The Computer Music Tutorial*, The MIT Press, 1996.

[16] Serra, X. "Musical Sound Modeling with Sinusoids plus Noise", *Musical Signal Processing*, Swets & Zeitlinger B. V., 1997.

[17] Steiglitz, K. A., *A Digital Signal Processing Primer, with Applications to Digital Audio and Computer Music*, Addison Wesley Publishing Company, Inc., 1996.

[18] Strawn, J. Editor, *Digital Audio Signal Processing: An Anthology*, William Kaufmann, Inc., 1985.

[19] Zölzer, U., Editor, *DAFX – Digital Audio Effects*. John Wiley & Sons, Ltd., 2008.

[20] Zölzer, U. *Digital Audio Signal Processing, Second Edition*, John Wiley & Sons, Ltd., 2008.