Doctoral Program In Music Technology

DARIUS PETERMANN
darius.petermann@gmail.com
dpetermann@berklee.edu

Advisor: Dr. Victor Lazzarini
Starting September 2016

An Overview of the Current Streaming Spectral Capabilities and Associated Parallel Computing and Research in Csound, and How It Relates To "Research in High Performance Computing Using Csound"

Using the Sliding Discrete Fourier Transform (SDFT) in Csound:

In Csound recently, an alternative to the fast Fourier transform (FFT) has been introduced – the sliding discrete Fourier transform (SDFT). Instead of jumping from frame to frame, as with the traditional FFT, with the SDFT, we slide from one frame to another. Consequently, we avoid the smearing associated with a traditional FFT. Further, the SDFT also implies that each bin can contain frequencies up to Nyquist, hence every bin can contribute to any frequency [4]. Thus, the SDFT used in spectral analysis/re-synthesis processing would allow extremely accurate results.

- In Csound, the SDFT can be realized by using pvsanal with an overlap of 1.
- Experimental CUDA opcodes [5] support the SDFT for analysis (using pvsanal with hopsize = 1) and re-synthesis purposes followed by minimal processing, with the current restriction on the type of DFT analysis window used; for example, the Kaiser window is not available. [4]
- A new and more efficient way of reconstructing f-signals has been devised in which each sample is represented by the current frame regardless of its size. [4]
- Currently the f-signal has its own rate in the Csound orchestra file. This rate should be equal to or larger than the k-rate. But, if the SDFT was optimized to support a larger palette of spectral processing opcodes, we could make use of a-rate variables on f-type signals to experiment with spectral processing

at a resolution equal to the sample rate.

- Several “verified” streaming spectral (PVS) opcodes already support the use of the SDFT. However, considering the number of streaming opcodes available in Csound [2], these “verified” SDFT opcodes [4] only represent a small portion of the streaming spectral processing potential currently available.
- Currently, the “verified” SDFT opcodes are very processor-intensive, and their ability to work properly depends both on specific NVIDIA hardware, and the capacity of the CUDA-based opcode library running on that hardware [6]. At the moment, opcodes that follow the CUDA parallel programming model are only available in source code and can only be built manually through CMake. Thus their study and use are quite limited.
- The CUDA opcodes look especially promising, because they have the potential to deliver extremely accurate spectral analysis and re-synthesis results - essentially giving each sample an analysis bin with a range from 0Hz to the Nyquist; and thus, they could lead to a number of significantly improved processing techniques such as zero latency
  Direct convolution.

As stated above, the SDFT is still a computationally expensive process, even considering the compute power of current hardware. A smarter implementation, using parallel SIMD hardware [7] for example, would allow real-time spectral processing through the use of SDFT, and this would open doors to new synthesis techniques, such as transformation in the spectral domain and we can expect many new sonic and musical possibilities to emerge.

Current State of High-Performance Computing (Hi-Pac) and Parallel Computing with Csound:

As John ffitch stated during the 2009 Linux Audio Conference, [8] and reinforced during a recent conversation, the speed increase related to processors is slowly coming to an end and yet, since we are always looking to extend and optimize our current palette of sounds, synthesis, and processing tools, researchers and musician needs to find new ways to make the best use of the current architectures available on consumer-grade hardware. Parallelism seems to be the key. In fact, ffitch further explained to me “the math are here [4] and ready to be implemented on a system that would serve parallel computing purposes.”

- Promising researches has been made [8], involving the use of parallel processing computing using multiple CPU multi-cores (from 2 to 8). In Csound [9], this would mean that the system would do most of its processing in parallel, in order to run multiple instances of instruments simultaneously, but according to ffitch, this is beneficial: “*as long as they do not use the same resource and as long as it is worth it!”* [8]
- Early researches on long recursive filters [10] shows that calculating an IIR filter in parallel, using multi processor architecture, could lead to a growth of speed without any additional latency. This is

particularly relevant because, until now, IIR systems have used no parallelism in the calculation process.

- Using the Python API [11] along with an enhanced build of the Python shell called IPython [12], live coding in Csound is now possible; but this process involves delegating the different rendering process associated with a particular Csound session, the score and orchestra, to multiple threads. Dispatching the whole rendering process on multiple threads allows the user to compile the Csound orchestra in parallel with the Csound score, thus avoiding the need to wait a whole k-rate cycle in order perform score events.

As mentioned before, it is no use to expect significant hardware, speed and capability improvements on general-purpose computers in the near future. What is being done in terms of parallel processing today is just the tip of the iceberg; but it show us the great potential lying in Csound and what might be possible if it was running on an optimal system. In their paper about the SDFT [7], Dobson, Bradford and ffitch mention that they present the SDFT as the "first of a new category of HiPAC processes", implying that there will be many more. I would like to devise some of them.

Csound Running Stand-alone, Under the Hood, as a Plugin, and on Portable Devices

Csound has a number of Front-Ends such as Cecilia, CsoundQt, Blue, CsoundForLive, and Cabbage; and these stand-alone programs, running with Csound inside, allow one to use Csound for real-time spectral processing. Cabbage can save its graphically wrapped Csound instruments as VST plugins and run them within many digital audio workstations and recording programs such as Audacity, Logic or Live. Via the csound~ external, Csound For Live runs Csound instruments directly in Ableton Live. Csound also runs on mobile devices such as the iPad, and on inexpensive microcontrollers such as the Raspberry Pi. Real-time spectral processing is possible on all these hardware and software systems and platforms, allowing musicians to exploit the power of Csound either as a plugin, to extend commercial software, or as a stand-along app, running Csound inside.

- The "Csound for iOS" API [13] made the implementation of Csound possible on iOS devices. apeSoft's *Sparkle*, Boulanger Labs' *csSpectral* [14] and my undergraduate thesis research at the Berklee College of Music - *csMorFFT* [15] are great examples of the real-time spectral processing potential of Csound on the iPad.
- The current PVS opcodes, tested and supported on iOS, are as follow:
  pvsadsyn, pvsanal, pvscross, pvsfread, pvsftr, pvsftw, pvsinfo, pvsmaska, pvsynth, pvstanal, pvsdiskin, pvscent,pvsdemix, pvsfreeze, pvsbuffer, pvsbufread, pvsbufread2, pvscale, pvshift, pvsifd, pvsinit, pvsin, pvsout, pvsosc, pvsbin, pvsdisp, pvsfwrite,pvslock, pvsmix, pvsmooth, pvsfilter, pvsblur, pvstencil, pvsarp, pvsvoc, pvsmorph, pvsbandp, pvsbandr, pvswarp, pvsgain, pvs2tab, tab2pvs [2]. Clearly, a wide palette of synthesis and processing techniques is already covered with these opcodes.

However, I could think of new spectral opcodes to be invented that would fit into this category including; spectral mutation, spectral mapping, spectral de-convolution.

- From the very beginning, Android devices would run Csound, but they suffered from severe and often prohibitive audio latency, making real-time processing out of the questions; but now, Android's "Gingerbread" eliminated the audio latency and makes this platform more suitable for real-time Csound work [16].
- Csound appears in many guises, boasts many front-ends, and its instruments and effects can even be compiled to work with commercial software, as *VST* or *AudioUnit* plugins [17]. Cabbage [18], a frontend allowing the user to create Csound-based plugins and standalone tools, includes many real time spectral-based instruments in its extensive instrument collection, and any of these intuitive and user-friendly graphical instruments can be run as VST plugins. Blue [19], a Csound graphical frontend, written in Java, allows for the design of a large palette of sound through the many synthesis and processing techniques that Csound offers, including spectral processing [20].
- The integration of Csound on embedded systems such as RaspberryPi 2 and BeagleBone [21] is widely used and some real-time spectral processing is possible at lower sampling rates or in mono.
- More recently the implementation of Csound on the Intel Galileo [22] have been made possible with the use of the Sysfs Linux interface; (some "small" adjustments to the library and SDK were necessary for Csound to run on these system). Custom frontends, gcsound1 for GEN1hardware, and gcsound2 for GEN2 hardware have been developed with the objective of making the implementation process easier and smoother for the user. As with all Arduino-like embedded systems, the board has digital IO and therefore some custom opcodes (gpin) have been developed so that Csound to take advantage of these on-board IO resources.

Running in real-time, and performing with Csound using such small portable devices with restricted flash memory and core processor capacity is the proof that we are only going forward in this field. However, spectral processing on these platforms is still quite computationally expensive for these smaller machines with less memory and slower processors. Thus, new and novel applications of Csound on these platforms still suffers from a lack of optimization that parallel processing could address. I plan to contribute in this area and on these platforms.

References

[1] Max Mathews, Joan Miller, F.R. Moore, *The Technology of Computer Music,* Chapter 3. "*Music V Manual",* The MIT Press, 1969

[2] Tools for Real-Time Spectral Processing (pvs Opcodes), http://www.csounds.com/manual/html/SpectralRealTime

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

[4] John ffitch, Richard Dobson and Russel Bradford, *"Sliding DFT for Fun and Musical Profit",* 2008

[5] Victor Lazzarini, Thomas Lysaght and Joseph Timoney, *"Streaming Spectral Processing With Consumer-Level Graphics Processing Unit",* DAFX-14 Conference, Erlangen, Germany September 1-5, 2014

[6] John ffitch, Steven Yi, and Victor Lazzarini. 2014. Csound on GitHub, Installing CUDA on Linux: https://github.com/csound/csound/wiki/Installing-the-CUDA-opcodes-in-Linux

[7] John ffitch, Richard Dobson & Russel Bradford, *"The Sliding Phase Vocoder",* Media Research Technology Center, Uni. Of Bath, UK

[8] John ffitch, Richard Dobson & Russel Bradford, *"The Imperative for High-Performance Audio Computing",* LAC2009, April 2009

[9] John ffitch & Martin Brain, *"Use of Multiple Core in Csound",* ICSC St. Petersburg, October 2015

[10] John ffitch & Richard Bradford, Unpublished Manuscript – in Progress and in Press *"Parallel Calculation for Long Recursive Filters",* Uni. Of Bath, 2016

[11] François Pinot, *"Real Time Coding using the Python API",* Csound Journal, Issue 14, Winter 2011

[12] Fernando Perez, Brian E. Granger, *"IPython: A System for Interactive Scientific Computing"* in "*Computing in Science and Engineering*", vol. 9, no. 3, pp. 21-29, May/June 2007

[13] Victor Lazzarini & Steven Yi, Csound for iOS Git Repo: https://github.com/csound/csound/tree/develop/iOS/

[14] Csound.Org, 2016, http://www.csounds.com/shop/csound-for-ios/

[15] Darius Petermann, *"A Deeper and More Extensive Look at Convolution",* http://www.darius-dsp.com/#!bachelor-thesis/jym09, March 2016

[16] Brian Redfern, *"Introducing the Android CSD Player; Jam Live with Android and Csound",* Csound Journal, Issue 17, Fall 2012

[17] Eddy Costello, Git Repo, Creating AU Plugins, https://github.com/eddyc/Csound-AudioUnit-Factory

[18] Rory Walsh, Git Repo, Cabbage, https://github.com/rorywalsh/cabbage

[19] Steven Yi, *"Blue; A Music Composition Environment for Csound",* http://blue.kunstmusik.com/

[20] Peiman Khosravi, *"On The Design of Spectral Tools in Blue",* Csound Journal, Issue 7, Fall 2007

[21] Paul Batchelor & Trev Wignall, *"An Introductory to Csound on the BeagleBone and RasberryPi, as Well as Other Linux-Powered Tinyware",* Csound Journal, Issue 18, Summer 2013

[22] Victor Lazzarini, Thomas Lysaght and Joseph Timoney, *"Embedded Sound Synthesis",* Maynooth University, 2011

[23] John ffitch, Steven Yi, and Victor Lazzarini. 2014. Csound on GitHub. http://csound. github.io.

[24] http://www.puredata.info/, March 2016

[25] http://www.audiosynth.com/, March 2016

[26] Victor Lazzarini, "*Sound Processing with The SoundObj Library",* DAFX-01 Conference, Limerick, Ireland December 6-8, 2001

[27] Dan Ellis, *"PVANAL File Format*", https://www.ee.columbia.edu/~dpwe/resources/pvanal.html, 1994

[28] Mark Dolson, "*The phase vocoder: A tutorial,*" Computer Music Journal, vol. 10, no. 4, pp. 14 -- 27, 1986.

http://www.panix.com/~jens/pvoc-dolson.par

[29] Richard Dobson, *"PVOC-EX: File Format for Phase Vocoder Data",*

http://people.bath.ac.uk/masrwd/pvocex/pvocex.html, May 2000

[30] Victor Lazzarini, Thomas Lysaght and Joseph Timoney, "*Spectral Signal Processing in Csound5",* Music Tech. Lab., National Uni. Of Ireland, Maynooth

[31] Darius Petermann, *"A Deeper and More Extensive Look at Convolution",* http://www.darius-dsp.com/#!bachelor-thesis/jym09, March 2016

[32] Victor Lazzarini, Thomas Lysaght and Joseph Timoney, *"Embedded Sound Synthesis",* Maynooth University, 2011

[33] Damian Keller, Victor Lazzarini, Marcelo S Pimenta, *"Ubiquitous Music",* Springer International Publishing, December 2014

[34] John ffitch, Richard Dobson & Russel Bradford, *"Sliding With a Constant Q",* DAFX-08 Conference, Espoo, Finland September 1-4, 2008

[35] Paris Smaragdis, *"System for Separating Multiple Sound Sources from Monophonic Signal with non-Negative Factor De-convolution"*, March 2004

[36] John ffitch, Richard Dobson & Russel Bradford, *"Transformation in the Spectral Domain" in "Sliding DFT for Fun and Musical Profit",* 2008

[37] *"CUDA C Programming Guide",* http://docs. nvidia.com/cuda/pdf/CUDA_C_Programming_ Guide.pdf, 2014.