Adaptive Whitening for Real-Time Onset Detection and Virtual Convolution of Impulsive, Iterative, Sonic Gestures

R. Michael Winters

December 15, 2011

Abstract

A patch for real-time onset detection and partitioned convolution was written in SuperCollider for the purpose of virtual convolution with a real sculpture. The project will be explained in terms of human echolocation and the developing field of sonic interaction design. Then, the problems of onset detection will be described as well as the approach and solution implemented currently. The author favours the use of palatal clicks as the primary sonic gesture for reasons that will be explained within the text.

1 Sonic Interaction Design and Gesture

Sonic Interaction Design (SID) is an emergent field interested in enhancing the role of sound in human-computer interaction. It draws on insights and developments in ubiquitous computing, auditory display, interaction design, and the interactive arts [1]. Although SID tends to focus on sound as an output medium, recent developments have emphasized the role of sound as input.

Although the input sounds that can be used will vary widely, a subset of those sounds are those that can be generated by the human body. [2] defines such sounds as *sonic gestures*, and to quote, "[A] sonic gesture [is] a sound-producing action generated by a human in order to convey information to a computational system." One such example of a sonic gesture in SID was an "eyes-free" user interface created using finger-snaps that were detected and localized in real-time [3]. The authors applied their algorithm to control three buttons of an MP3 player. Since then, hand clapping has been used to control music tempo [4], vowels to control mouse speed and direction (which actually competes with eye-tracking movement interfaces) [5], and humming and hissing for point and click tasks [6]. For a literature review of the most recent uses of sonic gesture, see section 2.1 of [2].

1.1 Convolution with a Virtual Sculpture

The patch was created in order to be used in an application for sonification of sculpture. In the paradigm, a blind user walks through an information field that varies with spatial location and directivity of the head. Wearing only headphones and a microphone, the user snaps, clicks, claps or taps, and information about the sculpture is sent to them vis sound. However, unlike a pure-delay, the sculpture information is convolved with the sonic gesture itself. It is as if the gesture enters an environment that can only be heard, ironically giving information that cannot be heard (although the human capacity for echolocation is astonishing) [7].

1.2 Design Considerations

When considering what sounds will be acceptable for real-time convolution in this application, it is important to consider them not only as researchers interested in design, but in terms of what types of sounds are actually the most useful to a blind person. A formal analysis of the physical and psychophysical properties of sounds used by the blind in echolocation tasks was taken performed in [8, 9]. They found that palatal clicks were the optimal pulses, and while the hand and finger produced pulses not lacking in quality, their relative position was more difficult to determine, less reproducible, and more subject to fatigue.

In a recent review of sonic gesture in SID, [2] characterizes gestures in terms of their temporal form (impulsive, sustained, iterative), and their extractable parameters (e.g. type, patterns, volume, deviation). Taken from this point of view, impulsive, iterative gestures are used in light of their use by the blind in echolocation. Although extractable parameters exist for these gestures, they are not used in the interaction.

2 Onset Detection

Using sonic gesture can involve detection, feature extraction, signal processing, and recognition, all in real-time. For this project, it involves onset detection with variable threshold and envelope tuned to minimal duration

(100ms). The majority of signal noise was removed through these two methods. The possibility that broadband noise in the form of speech or potentially wind (if outside) could weaken recoding quality was considered (see [10, 11, 12] for soultions). However, the need for real-time interaction, the variability of sound input, and the broadband characteristics of all but the palatal clicks made solutions that did not corrupt the input signal very transient.

2.1 Adaptive Whitening

Onset detection can be considered a subset of machine listening. Adaptive whitening was chosen as a preprocessing technique for onset detection in virtue of its published success [13] and it's availability in SuperCollider. It is doesn't require much processing power, can run in real-time, and can improve onset detection by as much as 10 percentage points. It runs on the current STFT frame in relationship to STFTs that preceeded it, making it a better choice for real-time than peak detection, which takes a whole extra STFT frame to compute. Especially in consideration of the fact that the beginnings of a finger snap last less than 2ms, one STFT frame really would have been too much.

Adaptive whitening is a procedure done on top of ordinary onset detection algorithms that may be idealized for a certain type of signal (e.g. drumbeats, melodies). It runs on the STFT as opposed to other methods such as filterbanks, wavelet decomposition, probabilistic modelling, or pitch tracking methods. In the paradigm, the peak-spectral-profile (PSP) is detected causally throughout the piece, but slowly decays proportional to a memory coefficient so that the previous peaks can be "forgotten." There is also a noise parameter so that the peak never goes below the noise floor. If it did, the noise would be emphasized and onset detection would be too sensitive.

2.2 Energy-Based Onset Detection Function

For the onset detection function (ODF), a simple energy-based detection function was used. The function is written [14]

$$D(i) = \sum_{j \in J} d_i(j) \tag{1}$$

where

$$d_i(j) = log_2 \left(\frac{|STFT_x^w(j,i)|}{|STFT_x^w(j,i-1)|} \right) \tag{2}$$

and

$$d_i = 1 \text{ if } d_i \geq T \tag{3}$$
$$d_i = 0 \text{ if } d_i < T \tag{4}$$

where i is the time index, j is the frequency bin, w is the window type, x is the hop size, and T is an experimentally determined threshold.

The choice to use energy-based detection came after informal experimentation with other common ODFs based upon phase-deviation, weighted phase-deviation, rectified spectral flux, complex deviation, and rectified complex deviation as implemented in [15], high-frequency component based ODF as implemented in [16], and modified kullback-leibler divergence as presented in [17]. The energy-estimation works really well for percussive inputs, much like the sounds that are the input parameters in the experiment. Furthermore, the energy based ODF is one of the least computationally expensive [13].

3 Implementation

The patch was written in SuperCollider 3.4 [18, 19, 20]. SuperCollider is a code-based audio-synthesis language unlike Max/MSP which is a visual programming language. As the chief collaborator Florian Grond was writing in SC, I thought it best to keep it in the same format. My experience in implementing this has been profound. I was able to quickly and easily implement advanced signal processing techniques and learn the inner workings of a computer in a way that I had not after a semester using Max/MSP. It was much less frustrating, and despite having relatively less experience, I felt a burden had been lifted.

3.1 Onset detection from a live microphone

First things first: in the patch, a function x is declared with internal variables *sig, localbuf, chain, onsets, env, snap.*

The input signal is the microphone signal amplified by a power of 10

$$sig = \text{SoundIn.ar(0,1);} \tag{5}$$

Our arguments are 0 and 1, the bus number and microphone volume respectively. I often found that magnitude amplification was necessary as the computer microphone often is not sensitive enough. However, when recording into a buffer, you can declare your mic input level which tends to avoid nasty feedback loops.

Within the function,

$$localbuf = \text{Buffer.alloc(s, 1024);} \tag{6}$$

allocates a buffer from the server of length 1024 samples. The buffer is local because it does not exist outside of the function x. This buffer is used to store the output from an fast-fourier transform that is implemented in the command

$$chain = \text{FFT(localbuf, sig);} \tag{7}$$

Chain is the variable which stores the info from our FFT output. Like fft in Max/MSP, the output of FFT is the real and imaginary parts of each spectral bin.

The variable *chain* is used as input to Onsets.kr. Onsets is a Unit generator (UGen) capable of onset detection of musical beats. The suffix .kr indicates that we are using the control rate as opposed to the audio rate (.ar). The control rate is useful for processes that do not require the speed of the audio rate. As a result, one calculation for every 64 samples is made, saving on processing power [20], which is definitely useful in real-time applications.

The command,

$$onsets = \text{Onsets.kr(}chain\text{, MouseX.kr(0,1), \textbackslash power);} \tag{8}$$

implements the adaptive whitening and the onset detection function \power (which is energy-based onset detection) as mentioned in section 2, while MouseX.kr(0,1); gives the user control over the threshold for detection via the horizontal position of the mouse. The arguments (0,1) imply that the left most mouse position is mapped to 0 and the right most mouse position is mapped to one. Usually a value between 0.4 and 0.5 works well. However, as the environmental noise context will change (as well as the desired way of listening), the user is allowed flexibility in determining the threshold. A lower threshold will trigger convolution kernels more often than a higher threshold. However, as a good rule of thumb, you should not be able to trigger the onset detector when the mouse is all the way at the right part of the screen (threshold=1). That allows a quick off switch.

For this part of the patch, the output of *snap* is first sent to a partition convolution function as discussed in section 3.3.

3.2 Storage in a buffer

I found that storage of this audio in a buffer was non-trivial, so I am including it in my report. As mentioned in section 2, this buffer could be used to try out noise-reduction techniques. However, whatever techniques are tried, it should be remembered that there are an infinite number of sonic gestures possible, so noise reduction must be general. Also, as one who has tried, remember that the sound-files are sensitive to distortion.

This part of the patch begins by declaring a global buffer "∼buf". "∼buf" is allocated from the server s and set to length 4500 samples. The number 4500 is chosen because this is approximately equal to

$$44100 \, \frac{\text{samples}}{\text{second}} * 0.102 \text{ second} \tag{9}$$

where $L = 0.102$ seconds is the the informal maximal length for any of the impulsive sonic gestures analyzed (i.e. hand clap, tongue click, finger snap, watch tap). This length was determined simply by recording each and analyzing the time-domain signal visually and aurally. The physical mechanics of each sound producing mechanism typically involve an initial impulse of length ≈ 3 ms followed by a resonance of $40 - 80$ ms. Sonograms of the four gesture types are displayed in section 4.

When Onsets.kr detects a beat, it sends a 1 as an output. For all other times, it sends 0 as output. The output of Onsets.kr is sent as a trigger for a redefined *snap* variable. The command

$$snap = \text{SoundIn.ar}(0,\text{EnvGen.kr}(env, onsets)); \tag{10}$$

uses the current sound input from the microphone as an input which has amplitude envelope characteristics controlled by the first argument of EnvGen.kr (*env*) and triggered by the second argument (*onsets*).

The first argument *env* is actually a variable that was declared at the beginning of our function. It is custom made envelope made with these gestures in mind. The variable *env* is defined

$$env = \text{Env.new}([0,1,1,0],[0.001,0.1,0.001],[0,0,0]); \tag{11}$$

which generates an box-like envelope that starts at 0 and reaches the max value of 1 after 0.001 seconds, holds 1 for 0.1 seconds, and releases back to 0 in 0.001 seconds. The slope for all segments is linear as indicated by the 0s in the third argument of Env, [0,0,0].

Finally, *snap* is recorded into the original buffer ∼*buf* using RecordBuf.ar which is triggered to start recording *snap* when *onset* is detected. In order to be able to analyze when a snap is recorded, the last part of the code

$$\text{Out.ar}(0,\ \text{Pan2.ar}(\mathit{sig},\ \text{-}1,\ 1) + \text{Pan2.ar}(\mathit{snap},\ 1,\ 1));\tag{12}$$

puts the microphone signal in the left ear and snap signal in the right ear. Sometimes, it is useful to just turn turn off the mic signal.

3.3 Partition convolution with a virtual environment

Whatever the signal is, it needs to be convolved with a virtual environment in real time. For our case, it will be a sculpture. This is not always a very efficient procedure and is computationally expensive. Fortunately, partitioned convolution allows for a great deal more efficiency [21]. Unfortuately, the sonification data was not prepared by the time of this report. To prepare it for eventual convolution, I quickly implemented PartConv with a virtual environment created using an algorithm by Dan Stowell [22]. By snapping ones finger (or anything else) you can hear it. It sounds like a big room.

3.4 Known Issues

By turning up the mic volume, you increase sensitivity, but as a result, sometimes your input will exceed the upper bounds of your buffer, creating distortions. If possible, you should keep the mic volume low and do any multiplication in the time domain signal.

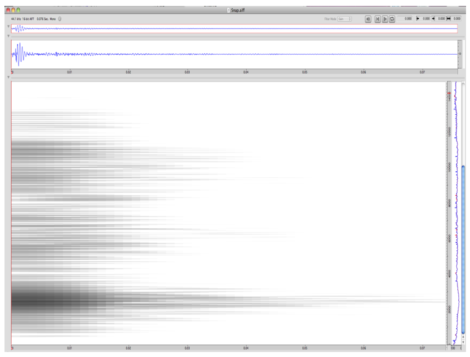
For recording a buffer for noise removal in real-time, you need to record output onto a buffer that will hold its contents yet rerecord when triggered again. This was not implemented out of lack of experience with SC.

3.5 Future Directions

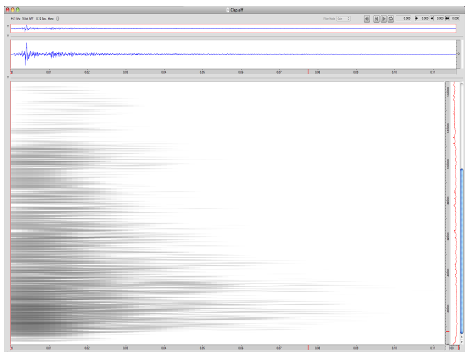
Research into human echolocation indicates that the use of palatal clicks are the most often used source for human echolocation in the blind. Reasons for this are discussed in section 1.2. However, given the SID paradigms all are useful in their own way, in particular for the parameters that can be extracted, I suggest that if microphones be put anywhere, they be put near the mouth in order to be able to use palatal clicks. Although time domain onset detection has proven useful for removing most noise, Fig. 1 discusses why palatal clicks may be candidates for further noise removal.

4 Sonograms of Four Sonic Gestures

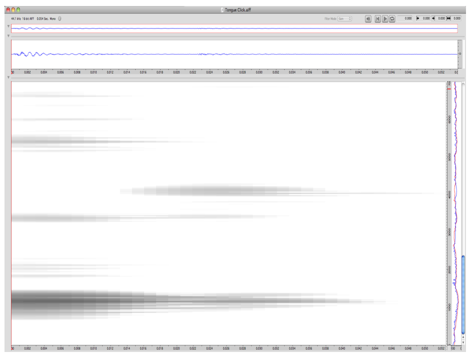
The four gestures were analyzed visually in AudioSculpt 2.9.4v3 using a sonogram FFT analysis with a Blackman-Harris window type, a window size of 2048 samples, and adaptive oversampling (32x). Recordings were made with an built in computer microphone.



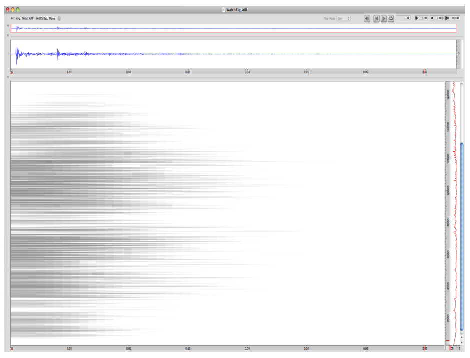
(a) Finger Snap



(b) Hand Clap



(c) Tongue Click



(d) Watch Tap

Figure 1: An array of four sonograms generated from four sonic gestures. All are possible in this paradigm, but the tongue click has been suggested [8] to be the most useful for the blind. For the others, the relative position is more difficult to determine, they are less reproducible, and more subject to fatigue. Although all are potential sonic gestures, the tongue click may be easier to remove from background broadband noise. All sounds displayed are less than 100ms long.

References

[1] D. Rocchesso, S. Serafin, F. Behrendt, N. Bernardini, R. Bresin, G. Eckel, K. Franinovic, T. Hermann, S. Pauletto, P. Susini, and Y. Visell, “Sonic interaction design: sound, information and experience,” in *CHI ’08 extended abstracts on Human factors in computing systems*, ser. CHI EA ’08. New York, NY, USA: ACM, 2008, pp. 3969–3972.

[2] A. Jyljä, “An eyes-free user interface controlled by finger snaps,” in *in Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05*, 2005, pp. 262–265.

[3] S. Vesa and T. Lokki, “An eyes-free user interface controlled by finger snaps,” in *in Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05*, 2005, pp. 262–265.

[4] A. Jylh and C. Erkut, “A hand clap interface for sonic interaction with the computer,” *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems CHI EA 09*, pp. 3175–3180, 2009.

[5] J. A. Bilmes, X. Li, J. Malkin, K. Kilanski, R. Wright, K. Kirchhoff, A. Subramanya, S. Harada, J. A. Landay, P. Dowden, and et al., *The Vocal Joystick: A voice-based human-computer interface for individuals with motor impairments.* Association for Computational Linguistics, 2005, no. October, p. 9951002.

[6] S. Chanjaradwichai, P. Punyabukkana, and A. Suchato, “Design and evaluation of a non-verbal voice-controlled cursor for point-and-click tasks,” in *Proceedings of the 4th International Convention on Rehabilitation Engineering & Assistive Technology*, ser. iCREATe ’10. Kaki Bukit TechPark II,, Singapore: Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, 2010, pp. 48:1–48:4.

[7] D. Kish, “World access for the blind,” http://www.worldaccessfortheblind.org/.

[8] J. A. M. Rojas, J. A. Hermosilla, R. S. Montero, and P. L. L. Espi, “Physical analysis of several organic signals for human echolocation: Han and finger produced pulses,” *Acta Acustica United with Acustica*, vol. 96, no. 6, 2010.

[9] J. A. M. A. M. Rojas, J. A. A. Hermosilla, R. S. S. Montero, and P. L. L. L. L. Espi, “Physical Analysis of Several Organic Signals for Human Echolocation: Oral Vacuum Pulses,” *Acta Acustica united with Acustica*, vol. 95, no. 2, pp. 325–330, 2010.

[10] R. Hoeldrich and M. Lorber, “Real-time broadband noise reduction,” *International Computer Music Journal*, 1998.

[11] S. J. Godsill and P. J. Rayner, *Hiss Reduction.* Berlin, Germany: Springer, 1998.

[12] ——, *Removal of Low Frequency Noise Pulses.* Berlin, Germany: Springer, 1998.

[13] D. Stowell and M. Plumbley, “Adaptive whitening for improved real-time audio onset detection,” *Proceedings of the International Computer Music Conference*, 2007.

[14] H. L. Tan, Y. Zhu, L. Chaisorn, and S. Rahardja, “Audio onset detection using energy-based and pitch-based processing,” in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 30 2010-june 2 2010, pp. 3689 –3692.

[15] S. Dixon, “Onset detection revisited,” in *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006, pp. 133–137.

[16] P. Masri, “Computer modeling for sound for transformation and synthesis of musical signals,” Ph.D. dissertation, University of Bristol, 1996.

[17] P. M. Brossier, “Automatic annotation of musical audio for interactive applications,” Ph.D. dissertation, University of London, August 2006.

[18] J. McCartney, *SuperCollider: a new real time synthesis language.* International Computer Music Association, 1996, pp. 257–258.

[19] SourceForge, “Supercollider: Real-time audio synthesis and algorithmic composition,” http://www.worldaccessfortheblind.org/.

[20] S. Wilson, D. Cottle, and N. Collins, Eds., *The SuperCollider Book.* Cambridge, Massachusetts: Massachusetts Institute of Technology, 2011.

[21] R. Battenberg, Eric ;Avizienis, “Implementing real-time partitioned convolution algorithms on conventional operating systems,” in *DAFX 2011*, Paris, France, 19/09/2011 2011.

[22] D. Stowell, “Dan stowell,” http://www.mcld.co.uk/.