MUSICAL APPLICATIONS OF NESTED COMB FILTERS FOR INHARMONIC RESONATOR EFFECTS

Jae hyun Ahn *Richard Dudas*
Center for Research in Electro-Acoustic Music and Audio (CREAMA)
Hanyang University School of Music
222 Wangsimni-ro Seongdong-gu
133-791 Seoul, South Korea

ABSTRACT

The comb filter is one of the basic building blocks in the world of digital filtering and signal processing, and an important component in a wide variety of musical and non-musical applications ranging from anti-aliasing of images and video to the design of numerous traditional audio effects. This paper describes a technique of using nested comb filter structures in order to design computationally stable inharmonic resonator effects with dynamically modifiable parameters. The number of parameters has additionally been kept to a minimum so that our inharmonic comb resonators can be easily and intuitively controlled for musical purposes. The comb filter implementations shown in this paper make use of new features of the Max/MSP 6 graphical programming environment, which allow both a more straightforward modification of and greater control over the low-level design of the comb filter algorithms themselves.

1. INTRODUCTION

The aim of this project was to provide users with a simple, easy-to-use inharmonic comb filter resonator effect that can be used for creative musical purposes. One important aspect in developing this effect was to explore comb filter design from a pedagogical point of view before embarking on the modification of traditional comb filter structures. Therefore, we decided to first create re-implementations of traditional comb filters alongside our final inharmonic comb filter, and provide them to users within the Max/MSP environment, making use of the gen~ object in Max 6. Naturally, the algorithms we present could also easily be rewritten for other signal processing languages and/or environments.

2. AN OVERVIEW OF COMB FILTERS

Comb filters are common tools in many realms of signal processing, from their use in television and telecommunications to their central role in many audio effects. Alongside other related filters, such as CIC (Cascaded Integrator Comb) filters (used extensively in analog to digital conversion), IIR (Infinite Impulse Response) filters and allpass filters, comb filters are not necessarily always used by themselves for musical purposes (although they can be), but are often integrated into more elaborate algorithms or used alongside other comb filters within larger signal processing applications. Such applications can range from simple audio effect scenarios such as echoes, chorus and flanging, to more complex realms of signal processing for artificial reverberation and physical modeling synthesis [6].

Although there are a wide variety of filter topographies for the internal structure of comb filters, and consequently just as many equations to describe them, they generally exist in two basic forms — feed-forward and feedback — which can either be used independently or merged together. When combined, the filters can either share a delay line, or be provided with separate delay lines (with independent delay times) for both the feed-forward and feedback stages.

2.1. Feed-Forward Comb Filters

A feed-forward comb filter delays the original input signal and sums it with the non-delayed signal at the output. The feed-forward coefficient, used directly as a multiplier for the delayed signal, can be used to create evenly-spaced *notches* in the output sound's spectrum, which become more pronounced as the coefficient nears unity gain. When this coefficient is negative, these notches start at DC and are spaced at harmonics of the frequency that corresponds to the delay time; when the coefficient is positive the notches occur at odd harmonics of half that frequency (thus creating an odd-harmonic spectrum one octave lower). Both cases are shown in figure 1.

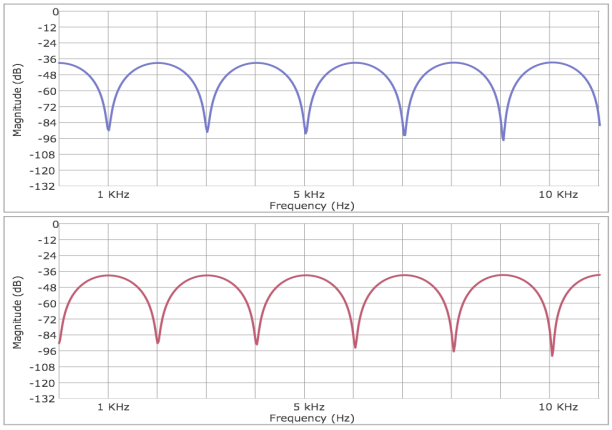

Figure 1. Frequency Response of a feed-forward comb filter for both positive (top) and negative (bottom) coefficient values and a delay time corresponding to a frequency of 2kHz (22.05 samples at a 44.1kHz SR).

The difference equation for a feed-forward comb filter (where a is the coefficient for the direct signal gain, b is the feed-forward coefficient and M is the delay time in samples) is:

$$y(n) = ax(n) + bx(n - M) \quad (1)$$

2.2. Feedback Comb Filters

A feedback comb filter has its delayed signal re-injected to a summing point just after the input and before the delay. The feedback coefficient (in this case used directly as a multiplier)[1] can be used to create *peaks* in the output sound's spectrum. In most implementations, when the feedback coefficient is positive these peaks start at DC and are evenly spaced at harmonics of the frequency corresponding to the delay time, and when the coefficient is negative the peaks occur at odd harmonics of half this frequency (once again outlining an odd-harmonic spectrum one octave lower), as shown in figure 2.

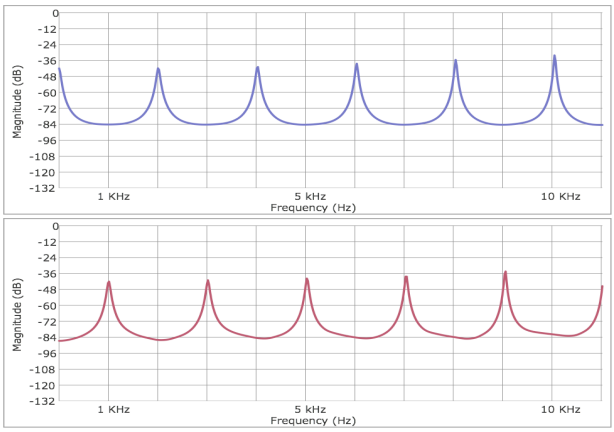

Figure 2. Frequency Response of a feedback comb filter for positive (top) and negative (bottom) coefficient values and a delay time corresponding to a frequency of 2kHz (22.05 samples at a 44.1kHz SR).

The difference equation for a feedback comb filter (where a is the coefficient for the direct signal gain, c is the sign-inverted feedback coefficient and M is the delay time in samples) is:

$$y(n) = ax(n) - cy(n - M) \quad (2)$$

2.3. Combined Comb Filter

The feed-forward and feedback comb filters can be combined together to form the standard comb filter found in most audio processing toolkits, whose signal flow block diagram is shown in figure 3. By combining the two we can obtain clearer peaks when the feed-forward and feedback coefficients are both positive or both negative, resulting in a harmonic comb spectrum when the coefficients are positive, or an odd-harmonic comb spectrum an octave lower when the coefficients are negative. Additionally, if the two coefficients have identical values with the opposite sign, the comb filter functions as an allpass filter, since the peaks and notches cancel each other out in amplitude. Naturally, in this case the filter's phase response remains complex.

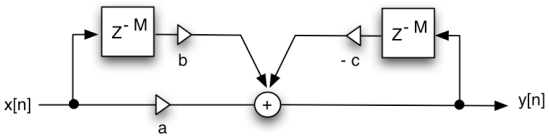

Figure 3. Block diagram for a standard comb filter combining feed-forward and feedback delays.

By rearranging the topography of the comb filter shown in figure 3 to use one shared delay that functions for both feed-forward and feedback, we can create a canonical comb filter similar in structure to Schroeder's allpass [1]. Although this topography, shown in figure 4, is not mathematically identical to the two-delay filter it is nonetheless equivalent and therefore produces the same filtering results as the comb filter with two delays shown in figure 3.

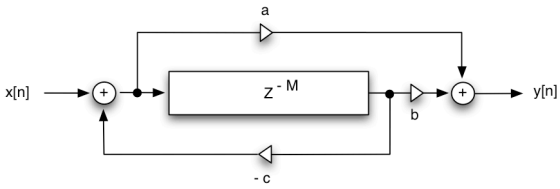

Figure 4. Block diagram for a canonical comb filter – this is an equivalent filter to that shown in figure 3, although its topography has been rearranged to share the delay line.

The filters shown in both of these signal flow diagrams can be described with following difference equation:

$$y(n) = ax(n) + bx(n - M) - cy(n - M) \quad (3)$$

It is worth noting that although many delay-based signal processing algorithms can be designed with either separate or shared delays, and can have their feed-forward placed either before or after the feedback loop, they are not always optimal for audio processing in all their forms [2]. In the case of the comb filter, both the two-delay and one-delay topographies shown here can be safely used for audio processing.

2.4. Allpass Lattice Filters

Allpass lattice filters, commonly used in waveguide-based physical modeling synthesis [8], have a similar signal flow structure to comb filters, albeit with a one-sample delay. Conveniently, a single coefficient, k, is used (converted into both positive and negative values for use within the algorithm) to control the phase response of the allpass filter. A first-order lattice section — not altogether very different in topography from the canonical comb filter presented above — is shown in figure 5.

[1] Most IIR comb filter implementations, including the ones shown here, invert the sign of the feedback coefficient, although some may not [7]. Inverting the sign of the coefficient means that the scaled feedback will be *added* to the direct signal, instead of being subtracted, as feedback generally is within digital filter algorithms.

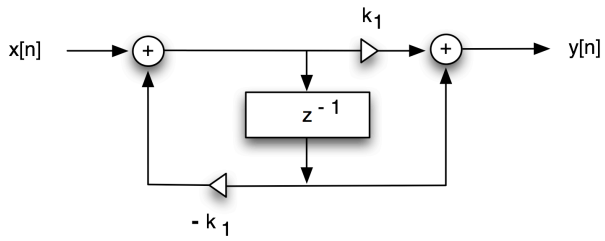

Figure 5. Block diagram for a direct form II first-order allpass lattice filter section.

Such first-order allpass filter sections can then be cascaded or nested (depending on the filter topography desired and/or the purpose for which they will be used) to produce higher order allpass lattice filters. A nested second order lattice filter is shown in figure 6. It is this nested structure that served as a starting point for our experimentation with nested comb filters. Note that the "inner" filter in the block diagram (highlighted with a dashed border) is nested in such a way that it simultaneously affects both the feed-forward and the feedback in the algorithm. Nested allpass structures used in other contexts, such as those used in some reverberation algorithms [3] are placed analogously.

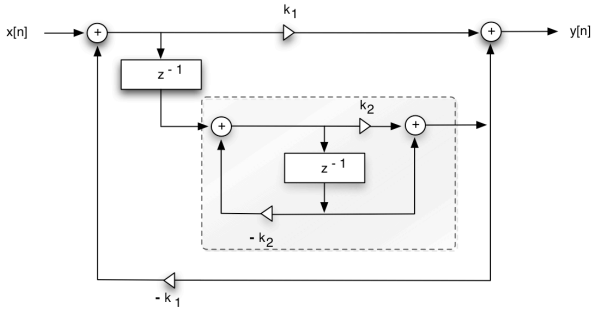

Figure 6. Block Diagram for a nested direct form II second-order allpass lattice filter.

3. REBUILDING COMB~ WITH GEN~

The Max/MSP environment comes with two standard comb filter objects integrated into the environment: comb~ and teeth~. The former is the classic comb filter effect, whose feed-forward and feedback delay lines are of equal lengths, whereas the latter allows different delay times for its feed-forward and feedback delay lines. These objects serve their function well, although they cannot be easily modified since they are compiled objects. The new gen~ object in Max 6 provides an environment of low-level signal processing tools that allow users to build their own recursive filters, delays, spectral processors, and sound generators (using 64-bit floating-point internal precision) in the same way as they would build a signal processing patch [10]. The advantage to the gen~ object is that the patch created inside it is compiled into efficient DSP code and therefore can be used to build a filter entirely identical to comb~ or teeth~.

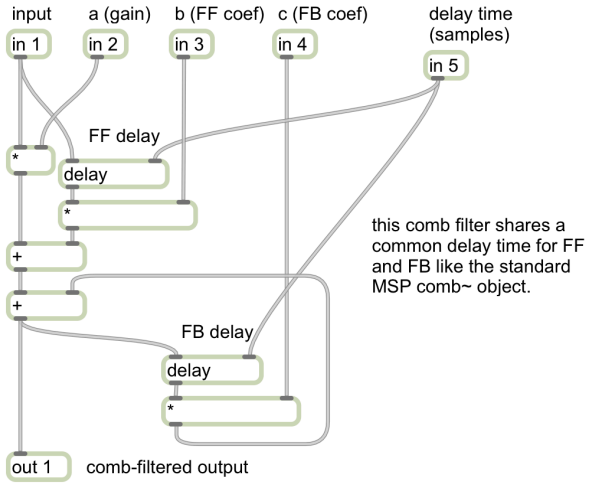

Figure 7. Max/MSP gen~ patch re-implementing the comb~ object. Note that the inlets are ordered slightly differently than those of the actual object and that we calculate the delay time in samples (based on a given frequency value) outside of this gen~ patch.

As a starting point, we built gen~ implementations of comb~ and teeth~ (both of which are a combination of a feed-forward and a feedback comb filter). The gen~ patch implementing comb~ is shown in figure 7; the implementation for teeth~ simply contains an extra inlet in order to be able to set the two delay times with different values. Comparing figures 3 and 7 you will notice that the gen~ patch representation actually slightly resembles the block diagram it implements.

Using gen~ also enabled us to build and test an equivalent version of comb~ using a shared delay topography, shown in figure 8. Again, it is useful to compare this implementation with its block diagram in figure 4.

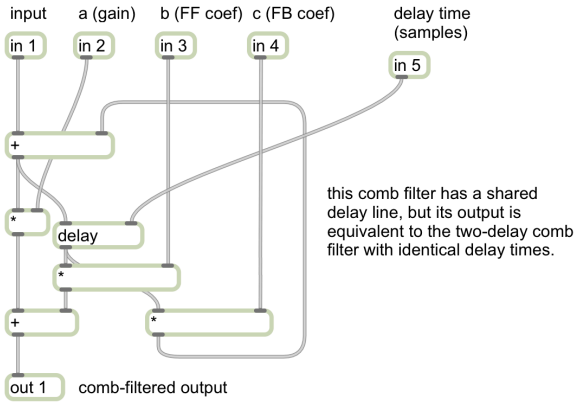

Figure 8. Max/MSP gen~ patch implementing a comb filter with a shared delay line.

By having a version of a basic comb filter (or any other object) in gen~, we can easily modify and extend it in unconventional ways. It occurred to us that nesting a comb filter inside the feedback loop of another comb filter, in much the same way that lattice filters can be nested, as mentioned above, could potentially produce interesting sonic results.

4. A NESTED COMB FILTER

It has already been shown that by inserting a meticulously designed high order allpass filter into the feedback loop of different types of resonator algorithms, "designer spectra" can be obtained [4] [5] [9]. This technique has been primarily used in the domain of physical modeling synthesis using digital waveguides. Nested Schroeder allpass structures have also been used successfully within reverberation algorithms alongside cascaded structures, in order to reduce unnatural sound coloration produced by the feedback loop(s) which simulate late reverberation [3].

4.1. Designing the Filter Algorithm

We initially discovered that we could obtain some interesting sonic results by nesting one comb filter within the feedback loop of another, or nesting two canonical comb filters in a way that resembled the Nested Direct Form II second-order lattice filters — whereby the inner comb filter is inserted into the signal chain just after the shared delay of the outer comb filter. However, many of our attempts at nesting filters required quite precise coefficient settings to work, as the resulting filter was often highly unstable and easily susceptible to "blowing up," especially when changing either of the two delay times. After empirically experimenting with some different filter topographies (one of the main advantages to using gen~ for this is that we can more easily visualize the signal flow than we could with text-based code), we discovered that we could obtain a stable filter structure by doing three things: 1) having equal and opposite multipliers for the inner (i.e., nested) comb filter, 2) having a shared multiplier for both the feed-forward and feedback loops of the outer comb filter, and 3) placing the inner (nested) filter *after* this multiplier. The use of equal and opposite multipliers for the inner comb filter renders it an allpass comb filter controlled by one coefficient, similar to the first order lattice filter, albeit with a multi-sample delay. The shared feed-forward/feedback multiplier for the outer filter simplifies controlling the resonant peaks of the comb. Finally, the placement of the inner filter means that it will affect both the feed-forward and feedback loops of the outer filter. The resulting comb filter is computationally stable and allows musically intuitive morphing between two sets of comb filters, passing through inharmonic spectra when moving between them.

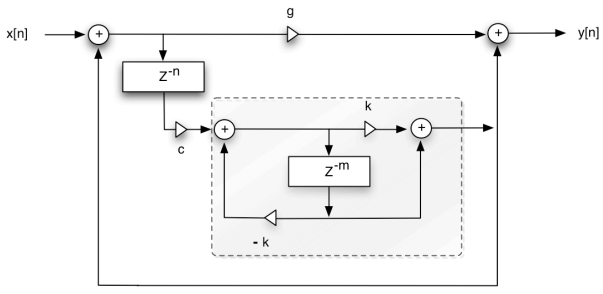

Figure 9. Block diagram showing the stable nested comb filter structure.

A signal flow block diagram for our nested comb filter is shown in figure 9. It can be described by the following set of difference equations, where *v(n)* is an intermediary calculation representing the point before the entry into the first delay, *w(n)* is the output point of the inner nester allpass comb, and the inner and outer delay lines have the lengths *M* and *N*, respectively:

$$v(n) = x(n) + cw(n)$$
$$w(n) = kv(n-N) + b(n-(M+N)) - kw(n-M) \quad (4)$$
$$y(n) = cw(n) + gv(n)$$

From a user's point of view, one important advantage to the nested comb filter topography we are using is that the number of coefficients is greatly reduced, providing control parameters not more complex than that of the teeth~ object (three coefficients and two delay times). Furthermore, and perhaps most importantly, there is a simple correlation between coefficient changes and audible results. The Max/MSP gen~ patch for this filter is shown in figure 10.

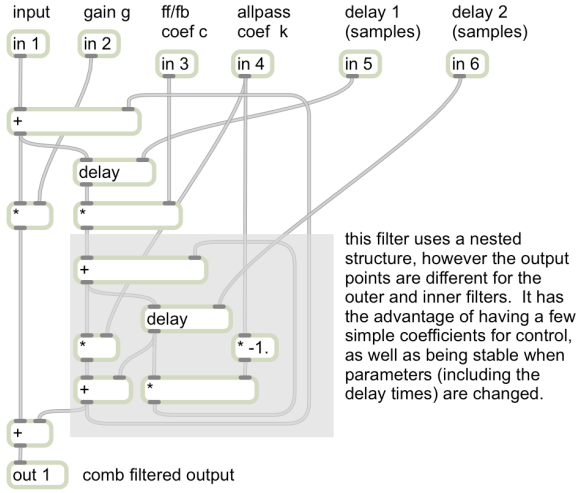

Figure 10. Max/MSP gen~ patch implementing the nested comb~ filter structure shown in figure 8. The delay times (in samples) are calculated outside this patch, based on two given frequencies.

The equivalent text-based code (which could be used directly in a codebox object within the gen~ patch) to describe the algorithm shown in figure 10 is as follows:

```
Delay delay_1(44100);
Delay delay_2(44100);
tap_3 = delay_1.read(in6);
mul_4 = in4 * -1.;
mul_5 = tap_3 * mul_4;
tap_6 = delay_2.read(in5);
mul_7 = tap_6 * in3;
add_8 = mul_7 + mul_5;
mul_9 = add_8 * in4;
add_10 = mul_9 + tap_3;
add_11 = in1 + add_10;
mul_12 = add_11 * in2;
add_13 = mul_12 + add_10;
out1 = add_13;
delay_1.write(add_8);
delay_2.write(add_11);
```

4.2. Results of the Filter Algorithm

This nested comb filter can produce two sets of equally spaced peaks based on frequencies corresponding to the delay time of the outer comb filter, and the sum of the delay time of both delay lines. Therefore, given any two desired principal resonant frequencies, f_1 and f_2, for the combs, the two delay times for the nested delay lines can be calculated as follows:

$$del_1 = \frac{SR}{\max(f_1, f_2)}$$
$$del_2 = \left| \frac{SR}{f_1} - \frac{SR}{f_2} \right| \quad (5)$$

Of our two resulting delay times, del_1 becomes the delay time in samples which used for the outer filter, and del_2 becomes the delay time in samples used for the inner (nested) filter. Presuming our input gain coefficient g is set to 1 and our k coefficient is set to 0, the c coefficient controls the peaks based on the lower of the two given frequencies (i.e., the frequency corresponding to the sum of both the delay times). If c is positive, the spectrum is harmonic, if it is negative, the spectrum is an odd-harmonic spectrum of half that frequency.

The coefficient k controls the multipliers for the inner (nested) comb filter (used here as an allpass) and can be varied between the limits of -1 to 1 (exclusive). It acts as an interpolator for the peaks of the comb filter as a whole. Presuming unity input gain and a positive value for c, if the value of k is zero the peaks are spaced at a frequency corresponding to the sum of the two delay times in samples (the lower of the two given frequencies). When k is near 1, the peaks are spaced in a harmonic spectrum at a frequency corresponding to the delay time of the outer comb filter (the higher of the two given frequencies); when k is near -1, the peaks are spaced in an odd harmonic spectrum half that frequency. If the c coefficient is negative the behaviour of the k coefficient will be inverted.

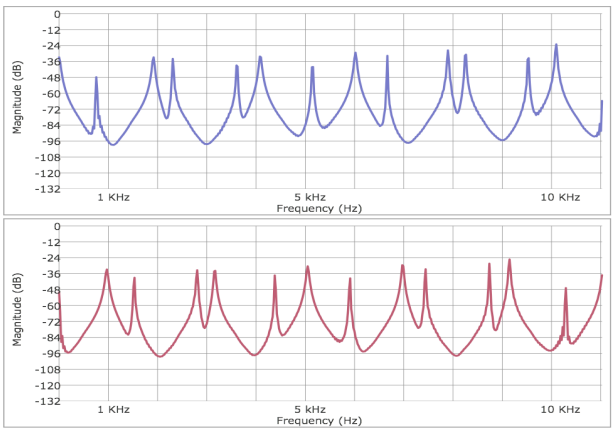

Figure 11. Frequency response of the inharmonic comb filter using delays corresponding to 2000Hz and 1470Hz, with g=1 and c=0.999. This graph shows positive and negative values for the interpolation coefficient: k=0.8 (top) and k=–0.8 (bottom).

Dynamically changing k causes the peaks to be both interpolated and cross-faded continuously from one comb spectrum to the other, producing inharmonic spectra similar to those obtained by frequency-shifting; it is in this region where the nested comb filter becomes most interesting, musically-speaking. An example frequency response for both positive and negative values of k is shown in figure 11.

The inharmonicity of the nested comb filter is naturally contingent upon the choice of frequency values as well as the value of k (see figure 12). In order to achieve a more linear perception of the interpolation between comb sets, we can use an arctangent function to compute k from a given linear value l between -1 and 1:

$$1 > x \geq 0 : k = \arctan\left(l^2 \cdot \tan(1)\right)$$
$$0 < x < -1 : k = -\arctan\left(l^2 \cdot \tan(1)\right) \quad (6)$$

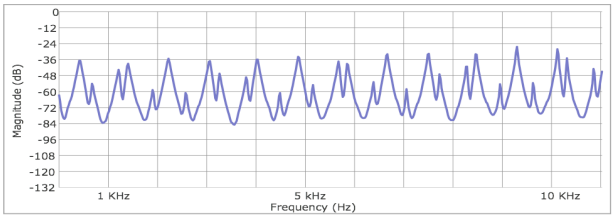

Figure 12. A representative inharmonic frequency response of the nested comb filter using delays calculated from the resonant frequencies 880Hz (A) and 370Hz (F#), with g=1 and c=–0.999 and k=0.743.

4.3. Additional Nested Filters

Although we did experiment with additional levels of nested filters, the results we obtained did not seem to offer any significant musical advantage over the nested pair of filters described above, neither in terms of interesting sonic results, nor computational stability during quick parameter changes. So, although this could be an interesting idea to explore at some point in the future, it did not seem to warrant our continuing in that direction at this time. Nevertheless, it goes without saying that more complex combinations of inharmonic resonators could be created by combining and arranging several of these singly-nested inharmonic comb filters in series or in parallel.

5. USES OF INHARMONIC COMB FILTERS

The nested "inharmonic comb filter" described here can be used in almost any context where regular harmonic comb filters are used. However they are particularly useful when used as resonator effects alongside percussion instruments, or "concrète" sound recordings of metallic objects. The main advantage of our nested comb filter structure is that it can be used to dynamically and stably interpolate between harmonic, odd-harmonic and inharmonic spectra, thereby easily and intuitively allowing musicians to "morph" between different sets of resonant timbres which can be defined by two predominant pitches within the context of a real-time performance situation.

We have already tested out our inharmonic comb filter in concert, in the context of a percussion piece by one of the authors, in order to provide inharmonic pitched resonance to the sound of predominantly non-pitched percussion instruments. We are providing information about this filter to the musical community in the form of a set of example patches in the Max/MSP environment and would like to encourage others to experiment with its use in creative contexts when it subjectively seems like it could be an appropriate tool for the musical task at hand.

6. CONCLUSION AND FUTURE WORK

This is an ongoing project, and we hope to develop other useful extensions of this filter in the future. From a composer's perspective, it would be nice to be able to provide a list of frequencies and be able to calculate the appropriate delay times and coefficients for a closely-matched inharmonic comb filter, instead of resorting to selecting the parameters empirically or by trial and error. In addition to the gen~ patches, we have also already created a preliminary version of the inharmonic comb filter as a standard compiled Max/MSP object. We are currently working on improving this object and enlarging our set of practical musical examples demonstrating this filter, in order to distribute them together with the gen~ patch shown above.

7. REFERENCES

[1] Cipriani, A., Giri, M., *Musica Elettronica e Sound Design, Teoria e Pratica con Max e MSP, Volume 2*, ConTempoNet s.a.s., Rome, Italy, 2013.

[2] Dattorro, J., "The Implementation of Recursive Filters for High-Fidelity Audio," *Journal of the Audio Engineering Society*, Vol. 36, No. 11, New York, NY, USA, 1988.

[3] Gardner, B., "A Realtime Multichannel Room Simulator," *Journal of the Acoustical Society of America*, Vol. 92, Issue 4, NY, USA, 1992.

[4] Jaffe, D. A., Smith, J. O., "Extensions of the Karplus-Strong Plucked-String Algorithm," *Computer Music Journal*, Vol. 7, No. 2, MIT Press, Cambridge, MA, USA, 1983.

[5] Karjalainen, M., Välimäki, V., Esquef, P. A. A., "Efficient Modeling and Synthesis of Bell-Like Sounds," *Proceedings of the 5th International Conference on Digital Audio Effects*, Hamburg, Germany, 2002.

[6] Roads, C., The Computer Music Tutorial, MIT Press, Cambridge, MA, USA, 1995.

[7] Smith, J. O., *Introduction to Digital Filters with Audio Applications*, W3K Publishing, USA, 2007.

[8] Smith, J. O., *Physical Audio Signal Processing for Virtual Musical Instruments and Digital Audio Effects*, W3K Publishing, USA, 2010.

[9] Van Duyne, S., Smith, J. O., "A Simplified Approach to Modeling Dispersion Caused by Stiffness in Strings and Plates," *Proceedings of the 1994 International Computer Music Conference*, Århus, Denmark, 1994.

[10] …, *Max 6 Reference Manual*, Cycling '74, San Francisco, California, USA, 2011.