LIVE CONVOLUTION WITH TIME-VARIANT IMPULSE RESPONSE

Øyvind Brandtsegg *

Department of Music,
Norwegian University of Technology and Science
Trondheim, Norway
oyvind.brandtsegg@ntnu.no

Sigurd Saue †

Department of Music,
Norwegian University of Technology and Science
Trondheim, Norway
sigurd.saue@ntnu.no

ABSTRACT

This paper describes methods for doing convolution of two live signals, without the need to load a time-invariant impulse response prior to the convolution process. It was developed in the context of creative live electronic music performance, but can be applied to more traditional use cases for convolution as well. The process allows parametrization of the convolution parameters, by way of real-time transformations of the IR, and as such can be used to build parametric convolution effects for audio mixing and spatialization as well.

1. INTRODUCTION

Convolution has been used for filtering, reverberation, spatialization and as a creative tool for cross-synthesis ([1], [2] and [3] to name a few). Common to most of them is that one of the inputs is a time-invariant impulse response (characterizing a filter, an acoustic space or similar), allocated and preprocessed prior to the convolution operation. Although developments have been made to make the process latency free (using a combination of partitioned and direct convolution [4]), the time-invariant nature of the impulse response (IR) has inhibited a parametric modulation of the process. Modifying the IR traditionally has implied the need to stop the audio processing, load the new IR, and then re-start processing using the updated IR.

2. CONTEXT

The current implementation was developed in the context of our work with cross-adaptive audio processing (see [5] and also the project blog http://crossadaptive.hf.ntnu.no/) and live processing (previous project blog at [6] and the released album "Evil Stone Circle" at [7]). Our previous efforts on live streaming convolution area are described in ([8], [9]). We also note that the area of flexible convolution processing is an active field of research in other environments (for example [10], in some respects also [11] and [12] due to the parametric approach to techniques closely related to convolution).

Our primary goal has been to enable the use of convolution as a creative tool for live electronic music performance, by allowing two live sources to be convolved with each other in a streaming fashion. Our current implementation allows this kind of live streaming convolution with minimal buffering. As we shall see, this also allows for parametric transformations of the IR. Examples of useful transformations might be pitch shifting, time stretching, time reversal, filtering, all done in a time-variant manner.

* Thanks to NTNU for generously providing a sabattical term, within which substantial portions of this research have been done

† NTNU: this guy needs more time to do undisturbed research

3. PREVIOUS IMPLEMENTATIONS

Convolution of two finite sequences *x(n)* and *h(n)* of length N is defined as:

$$y(n) = x\,(n) * h\,(n) = \sum_{k=0}^{N-1} h\,(k)\,x\,(n-k) \qquad (1)$$

This is a direct, time-domain implementation in a FIR filter structure with filter length N. A few observations can be made at this stage:

- There are no parameters involved.
- There is no latency: $y(0) = x(0)h(0)$.
- Output length is $N_y = N_x + N_h - 1 = 2N - 1$
- Computational complexity is of the order $O(N^2)$.

It should also be noted that the convolution output exhibits a very characteristic time smearing. The computational complexity is prohibiting with the segment sizes we normally work with (1-2 seconds or more). A far more efficient solution is fast convolution using FFT and multiplication in the frequency domain [13]:

$$\mathcal{F}[x(t) * h(t)] = \mathcal{F}[x(t)] \cdot \mathcal{F}[y(t)] = X(f) \cdot H(f) \qquad (2)$$

This implementation also calls for some observations:

- Computational complexity is reduced to $O(N \log N)$.
- Latency has increased to the full filter length.

It is now obvious that convolution is simply multiplication in the frequency domain. From which we may further observe that:

- Output amplitude strongly depends on degree of frequency overlap between the two inputs
- We may expect a relative loss of high frequency energy.

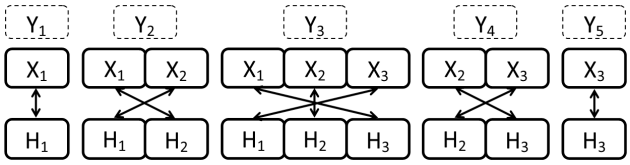

Figure 1: *Example of partitioned convolution with 3 partitions [8]*

The increased latency is undesirable for real-time applications. *Partitioned convolution* reduces the latency by breaking up the input signal into smaller partitions [14]. A trivial example of the computation with only 3 partitions is shown in Figure 1. Partition size should be adjusted to make an optimal compromise between computational efficiency and latency.

From this implementation we observe:

- Latency is reduced to the partition size
- The first partition output Y_1 depends only on X_1 and H_1
- Partition output Y_{N+1} does not depend on X_1 and H_1, output Y_{N+2} does not depend on X_2 and H_2, and so on.

The latter two observations play an important role for our streaming approach. It should also be added that techniques combining partitioned and direct-form convolution can eliminate processing latency entirely [4].

Traditionally convolution is an asymmetric operation where the two inputs have different status: input signal and impulse response (IR) respectively. Typically the latter is a reasonably short segment, a time-invariant representation of a system (e.g. a filter or a room response) onto which the input signal is applied. Hence the impulse response in equation 1 can be formulated using constant scaling coefficients a_0 to a_{N-1}:

$$y(n) = x\,(n) * h\,(n) = \sum_{k=0}^{N-1} a_k x\,(n-k) \tag{3}$$

Instead we are searching for more flexible tools for musical interplay and have previously presented a number of strategies for dynamic convolution [9]. Our aim has been to:

- Attain dynamic parametric control over the convolution process in order to increase playability.
- Investigate methods to avoid or control dense and smeared output
- Provide the ability to update/change the impulse responses in real-time without glitches.
- Provide the ability to use two live audio sources as inputs to a continuous real-time convolution process.

With existing tools we haven't been able to get around the time-invariant nature of the impulse response. In order to make dynamic updates of the IR during convolution without audible artifacts, we had to use two (or more) concurrent convolution processes and then crossfade between them whenever the IR should be modified. The IR update could be triggered explicitly, at regular intervals or based on the dynamics of the input signal (i.e. transient detection). Every update of the IR triggered a reinitialization of the convolution process.

We have also done work on live convolution of two audio signals of equal status: neither signal is the IR of the other [8]. Instead both audio streams are segmented at intervals triggered by transient detection and each pair of segments convolved in a separate process. With frequent triggering the number of concurrent processes could grow substantially due to the long convolution tail (N-1 partitions) of each process.

4. OUR IMPLEMENTATION: TIME-VARIANT IMPULSE RESPONSE

In this paper we present a simple, but efficient implementation of convolution with a time-variant impulse response. In this case the coefficients of the IR are no longer constants (ref equation 3), but are themselves depending on the time variable n:

$$y(n) = x\,(n) * h\,(n) = \sum_{k=0}^{N-1} a_k(n) x\,(n-k) \tag{4}$$

The goal is to be able to dynamically update the impulse response without reinitializations and crossfades of parallel convolution processes. The key to our approach is the previously noted property of partitioned convolution:

> An output partition Y_n only depends on input partitions X_k and H_k for $n, k \geq 1$ and $k \in [n+1-N, n]$, where N is the number of partitions of the impulse response.

An immediate consequence of this property is that we can load the impulse response partition by partition in parallel with the input. A fully loaded IR is not necessary to initiate the convolution process. This drastically reduces the latency for application of live sampled impulse responses. In addition the computational load associated with FFT calculations on the IR is nicely spread out in time, hence avoiding bursts of CPU usage when loading long impulse responses. The only actions necessary during initialization of the convolution algorithm, are to allocate memory for the chosen IR length, clear buffers and initialize variables.

The opposite also holds true: We can *unload* the impulse response partition by partition without audible artifacts, even while the convolution process is running. Unloading a partition simply means clearing it to zero, and it should progress in the same order as the loading process. The shortest possible load/unload interval is a single partition, which is actually equivalent to segmenting out a single partition of the *input signal* and convolve it with the full impulse response. Figure 2 illustrates the effect.

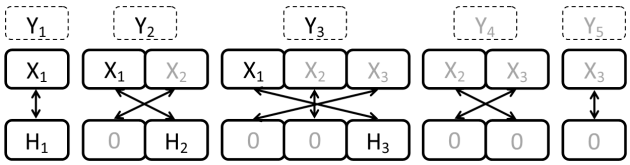

Figure 2: *Example of minimal load/unload interval*

The combination of the two strategies, stepwise loading and unloading, naturally leads to *stepwise replacement* of the impulse response. At any time during the running convolution process we can trigger an update of the IR and start filling in new partitions from the beginning of the IR buffer. Figure 3 shows a simplified example of the procedure. The impulse response $H_{1,k}$ is replaced by $H_{2,k}$. In a transition period equal to $N-1$ (where N is the IR length) the output is a mix of two convolution processes.

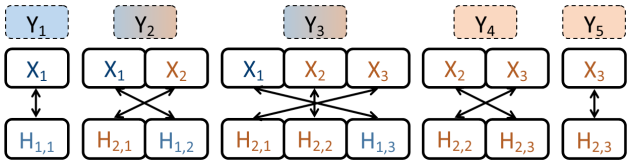

Figure 3: *Example of dynamic IR update*

It should be apparent that triggering a new IR update is also a segmentation of the input signal. No input partitions buffered before the trigger will be convolved with the new IR, and similarly, no input partitions buffered after the trigger will be convolved with the old IR. To avoid discontinuities in the IR and hence audible clicks in the output signal, the impulse response buffer should always be completely refilled when updated (all N partitions).

We have implemented convolution with time-variant IR as a plugin opcode titled *liveconv* in the audio programming language Csound[1]. A Csound opcode normally provides three programming components: An internal data structure, an initializing function and a processing function [15]. The internal data structure is allocated in the initialization function, including all dynamic memory. In *liveconv* partition length and impulse response table are specified during this step. The content of the IR table may be updated at any time, but not its memory location or size.

The outline of the processing function is as follows (details on overlap-add are omitted):

- Check the update control signal. If set to "load", prepare to reload the IR. If set to "unload", prepare to unload the IR. A data structure maintains control of each load/unload process. In theory there could be a new process for every partition.
- Read audio inputs into an internal circular buffer in chunks given by the control rate of Csound (*ksmps* is the number of samples processed during each pass of the processing function).
- When an entire partition is filled with input data:
  - Calculate the FFT of the input partition
  - For every running *load* process fetch a specified partition from the IR table and calculate its FFT. Note that several parts of the IR may be updated in parallel.
  - Do complex multiplication of input and IR buffers in the frequency domain
  - Calculate the inverse FFT of the result and write to output
  - For every running *unload* process clear a specified partition to zero.
- Increment load/unload processes to prepare for the next partition. If a process has completed its pass through the IR table, it is set inactive.

No assumptions have to be made on the impulse responses involved. The load process can be signaled as soon as one *ksmps* chunk of new data has been filled into the IR table.

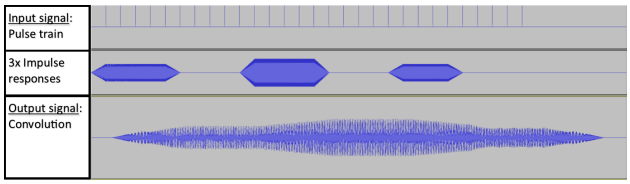

Figure 4: *Example of dynamic convolution with pulse train as input signal and three different impulse responses.*

Figure 4 shows a simple example of convolution between a pulse train and three different impulse responses. The three IRs are placed at their point of triggering. Notice how the output amplitude gradually transforms for each new IR.

A special case appears when the IR's are sampled from the same continuous audio stream, one partition apart. Then we get a result similar to cross-synthesis, pairwise multiplication of partitions from two parallel audio streams, but still with the effect of long convolution filters. It should be added that there are more efficient ways to implement this particular condition, but that will be deferred to another paper.

The interval between IR updates is provided as a parameter available for performance control, which increases the playability of convolution. There is still a risk of huge dynamic variations due to varying degrees of frequency overlap between input signal and impulse response. The housekeeping scheme introduced to maintain the IR update processes could be exploited for smarter amplitude scaling. This is ongoing work. We would also like to look at integration of some of the extended convolution techniques proposed by Donahue & al [10].

5. USE CASE: PLUGIN FOR LIVE PERFORMANCE

A dedicated VST plugin has been implemented to show the use of the new convolution methods, built around the *liveconv* opcode. Even though the incremental IR update allows for a vast array of application areas, we have initially focused on realtime convolution of two live signals. As an indication of its primary use, we've named it "Live convolver". The plugin is implemented with a "dual mono" input signal flow, as shown in figure 5, allowing it to be used on a stereo track of a DAW. The *left* stereo signal will be used to record the IR, while the *right* stereo signal will be used as input to the convolution process.

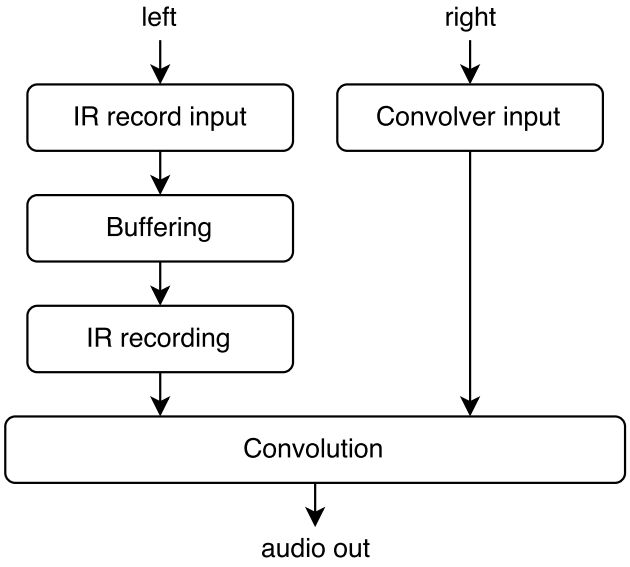

Figure 5: *Plugin signal flow overview*

The signal on the *IR record input* channel is continuously written to a circular buffer. When we want to replace the current IR, we read from this buffer and replace the IR partition by partition. The main reason for the circular buffer is to enable transformation (for example time reversal) of the audio before making the IR. As an attempt to visualize *when* the IR is taken from, we use a circular coulouring scheme to display the circular input buffer. We also represent the IR using the same colours. Time (of the input buffer) is thus represented by colour. As the color of *now* continuously

[1] More information at http://csound.github.io/

and gradually changes from red to green to blue, it is possible to identify *time lag* as an azimuthal distance on the color circle [2]. Figure 6 shows the plugin gui, with a representation of the coloured input buffer and a live sampled IR. Similarly, a time reversed IR is shown in figure 7. Note the direction of color change (green to red) of the reversed IR as compared with the color change in the normal (forward) IR (blue to red).

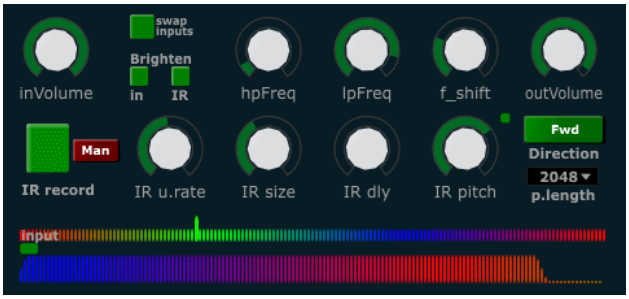

Figure 6: *Liveconvolver plugin GUI*

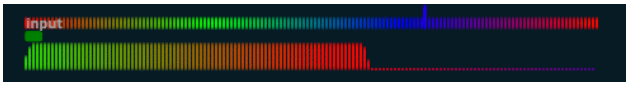

Figure 7: *Visualization of time reversed IR*

The plugin has controls for manual triggering of the IR recording, or the IR can be automatically updated by a periodic trigger (using *IR update rate* and *IR size controls*). Methods for triggering IR update via transient detection has also been implemented. Pitch modification and time reversal is available by dedicated gui controls. In addition, we have simple lowpass and highpass filtering, as this can be very useful for quickly fixing spectral problems of using convolution in a live setting. Since convolution can lead to a certain loss of high frequency content, we also have the option of "brightening" each input by applying a low-q high shelf filter. Finally, there is a significant potential for audio feedback using convolution in a live setting, when the IR is sampled in the same room where the convolver output is played back. To alleviate this risk, we have implemented a frequency shifter with relatively small amounts of shift as suggested by [16]. The amount of frequency shift is also controllable from the gui. By shifting the convolver output with just a few Hz, the feedback potential is significantly reduced.

6. USE CASE: LIVE PERFORMANCES

The liveconvolver plugin has been used for several studio sessions and performances from late 2016 onwards. Reports from some of these experimental sessions can be found at [17] and [18]. These reports also contain reflections on the performative difference experienced in the roles of *recording the IR* and *playing through it*. It is notable that a difference is perceived, since in theory the mathematical output of the convolution process is identical regardless of which one signal is used as the IR. The buffering process allows continuous updates to the IR with minimal latency, and several methods for triggering the IR update has been explored. This should ideally facilitate a seamless merging of the two input signals. However, the IR update still needs to be triggered, and the IR will be invariant between triggered updates. In this respect, the instrument providing the source for the IR will be subject to recording, and the live input to the convolution process is allowed a continuous and seamless flow. It is natural for a performer to be acutely aware of the distinction between being recorded and playing live. Similar distinctions can assumably be made in the context of all forms of live sampling performance, and in many cases of live processing as an instrumental practice. The direct temporal control of the musical energy resides primarily with the instrument *playing through the effects processing*, and to a lesser degree with the instrumental process *creating the effects process* (recording the IR in our case here).

[2] https://en.wikipedia.org/wiki/Color_wheel

7. OTHER USE CASES, FUTURE WORK

The flexibility gained by our implementation allows parametric control of convolution also in more traditional effects processing applications. One could easily envision a parametric convolution reverb with continuous pitch and filter modulation for example. Parametric modulation of the IR can be done either by applying transformations on the audio being recorded to the IR, or directly on the spectral data. Such changes to the IR could have been done with traditional convolution techniques too, by preparing a set of transformed IR's and crossfading between them. The possibilities inherent in the incremental IR update as we have described allows direct parametric experimentation, and thus is much more immediate. It also allows for automated time-variant modulations (using LFO's and random modulators), all without introducing artifacts due to IR updates.

8. CONCLUSIONS

We have shown a technique for incremental update of the impulse response for convolution purposes. The technique provides time-variant filtering by doing a continuous update of the IR from a live input signal. It also opens up possibilities for direct parametric control of the convolution process and as such enhancing the general flexibility of the technique. We have implemented a simple VST plugin as proof of concept of the live streaming convolution, and documented some practical musical exploration of its use. Further applications within more traditional uses of convolution has also been suggested.

9. ACKNOWLEDGMENTS

We would like to thank the Norwegian Artistic Research Programme for support of the research project "Cross-adaptive audio processing as musical intervention", within which the research presented here has been done. Thanks also to the University of California, and performers Kyle Motl and Jordan Morton for involvement in the practical experimentation sessions.

10. REFERENCES

[1] Mark Dolson, "Recent advances in musique concrète at CARL," in *Proceedings of the 1985 International Computer Music Conference, ICMC 1985, Burnaby, BC, Canada, August 19-22, 1985*, 1985.

[2] Curtis Roads, "Musical sound transformation by convolution," in *Opening a New Horizon: Proceedings of the 1993 International Computer Music Conference, ICMC 1993, Tokio, Japan, September 10-15, 1993*, 1993.

[3] Trond Engum, "Real-time control and creative convolution," in *11th International Conference on New Interfaces for Musical Expression, NIME 2011, Oslo, Norway, May 30 - June 1, 2011*, 2011, pp. 519–522.

[4] William G. Gardner, "Efficient convolution without input-output delay," *Journal of the Audio Engineering Society*, vol. 43, no. 3, pp. 127, 1995.

[5] Øyvind Brandtsegg, "A toolkit for experimentation with signal interaction," in *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, 2015, pp. 42–48.

[6] Øyvind Brandtsegg, Trond Engum, Andreas Bergsland, Tone Aase, Carl Haakon Waadeland, Bernt Isak Wærstad, and Sigurd Saue, "T-emp communication and interplay in an electronically based ensemble," https://www.researchcatalogue.net/view/48123/48124/10/10, 2013.

[7] Øyvind Brandtsegg, Trond Engum, Tone Aase, Carl Haakon Waadeland, and Bernt Isak Wærstad, "Evil stone circle," https://www.cdbaby.com/cd/temptrondheimelectroacou, 2015.

[8] Lars Eri Myhre, Antoine H Bardoz, Sigurd Saue, Øyvind Brandtsegg, and Jan Tro, "Cross convolution of live audio signals for musical applications," in *International Symposium on Computer Music Multidisciplinary Research*, 2013, pp. 878–885.

[9] Øyvind Brandtsegg and Sigurd Saue, "Experiments with dynamic convolution techniques in live performance," in *Linux Audio Conference*, 2013.

[10] Chris Donahue, Tome Erbe, and Miller Puckette, "Extended convolution techniques for cross-synthesis," in *Proceedings of the International Computer Music Conference 2016*, 2016, pp. 249–252.

[11] Jonathan S Abel, Sean Coffin, and Kyle S Spratt, "A modal architecture for artificial reverberation," *Journal of the Acoustical Society of America*, vol. 134, no. 5, pp. 4220–4220, 2013.

[12] Jonathan S Abel and Kurt James Werner, "Distortion and pitch processing using a modal reverberator architecture," in *International Conference on Digital Audio Effects (DAFx-15)*, Trondheim, Norway, November/2015 2015, .

[13] H.J. Nussbaumer, *Fast Fourier Transform and convolution algorithms*, Springer, 1982.

[14] Thomas G. Stockham, "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring joint computer conference*, 1966, pp. 229–233.

[15] Victor Lazzarini, "Extensions to the csound language: from user-defined to plugin opcodes and beyond," in *Proceedings of the 3rd Linux Audio Conference*, 2005.

[16] Carlos Vila, "Digital frequency shifting for electroacoustic feedback suppression," in *Audio Engineering Society Convention 118*, May 2005.

[17] Øyvind Brandtsegg and Kyle Motl, "Session ucsd 14. februar 2017," http://crossadaptive.hf.ntnu.no/index.php/2017/02/15/session-ucsd-14-februar-2017/, 2017.

[18] Øyvind Brandtsegg and Jordan Morton, "Convolution experiments with jordan morton," http://crossadaptive.hf.ntnu.no/index.php/2017/03/01/convolution-experiments-with-jordan-morton/, 2017.