A NEW REVERBERATOR BASED ON VARIABLE SPARSITY CONVOLUTION

Bo Holm-Rasmussen\textsuperscript{a,b}  
\textsuperscript{a}Acoustic Technology  
Department of Electrical Engineering  
Technical University of Denmark  
Kgs. Lyngby, Denmark  
\textsuperscript{b}Department of Signal Processing and Acoustics  
School of Electrical Engineering  
Aalto University  
Espoo, Finland

Heidi-Maria Lehtonen\textsuperscript{b} and Vesa Välimäki\textsuperscript{b}

ABSTRACT

An efficient algorithm approximating the late part of room reverberation is proposed. The algorithm partitions the impulse response tail into variable-length segments and replaces them with a set of sparse FIR filters and lowpass filters, cascaded with several Schroeder allpass filters. The sparse FIR filter coefficients are selected from a velvet noise sequence, which consists of ones, minus ones, and zeros only. In this application, it is sufficient perceptually to use very sparse velvet noise sequences having only about 0.1 to 0.2\% non-zero elements, with increasing sparsity along the impulse response. The algorithm yields a parametric approximation of the late part of the impulse response, which is more than 100 times more efficient computationally than the direct convolution. The computational load of the proposed algorithm is comparable to that of FFT-based partitioned convolution techniques, but with nearly half the memory usage. The main advantage of the new reverberator is the flexible parameterization.

1. INTRODUCTION

Artificial reverberation has been a popular audio effect since the early studio recordings almost a century ago. Rooms and halls are considered being both linear and time-invariant (LTI) systems, regarding sound in the audible range. Therefore the sonic characteristic of a specific concert hall can be replicated by an LTI system having the same impulse response. Impulse responses of concert halls normally contain three important phases: the direct (a.k.a. dry) sound, the early reflections, and the late reverberation which has dense reflections [1].

Many artificial digital reverberation algorithms have been described since Schroeder and Logan published the first one in the early 1960s [2, 3]. The realtime algorithms are often categorized as based on either delay networks, convolution or a hybrid [1]. Algorithms based on delay networks are comprised of delays, filters, and feedback paths and they are characterized by low memory requirements, low computational complexity, and good parameterization but often lacking realism. The original algorithms by Schroeder and Logan, which were subsequently improved by Moorer [4], and the feedback delay networks (FDN) originally proposed by Jot and Chaigne are popular examples of delay network algorithms [5, 6, 7, 8].

Convolution by the desired room impulse response (RIR) results in very realistic reverberation but is difficult to parameterize; the partitioned fast convolution method reduces the computational complexity considerably compared to direct (FIR filter) convolution and avoids the delay introduced by full-length FFT convolution [9, 10]. A recent article by Välimäki \textit{et al.} provides a thorough overview of the development of artificial reverberation [1].

In this paper a novel hybrid late reverberation algorithm based on convolution, implemented as sparse FIR (SFIR) filters, and optimized with delay network elements is presented. The SFIR coefficients are extracted from a specific kind of white noise known as velvet noise [11, 12]. Schroeder allpass (SAP) filters [2] are used to allow SFIR filters with greater sparsity to be used. The motivations for this work have been the desire to find a computationally efficient and flexible reverb algorithm, which would be easy to control and to calibrate to a recorded RIR.

The SFIR filters in the proposed algorithm can be seen as an unwrapped FIR version of the late reverberation algorithms based on a single SFIR filter and a feedback loop first proposed by Rubak and Johansen [13, 14]. Later Karjalainen and Järveläinen refined the algorithm both sonically and in terms of lower computational complexity by using sparse coefficients extracted from velvet noise, and by continuously updating the sparse coefficients [11]. Recently Lee \textit{et al.} [15] have proposed several ways of improving the sonic qualities of this kind of algorithm further, mainly by how the update of the sparse coefficients takes place. The sonically most promising of these algorithms sounds reasonably good for most input signals but the performance of the algorithm is difficult to foresee because of its time variance. Moreover, it can only replicate RIRs with strictly exponentially decaying reverberation times because the filter attenuator is in a feedback loop.

The proposed reverberator provides high-quality zero-delay diffuse late reverberation sonically comparable to convolution and with similar computational complexity, latency, and approximately half the memory requirements compared to the partitioned fast convolution. The spectral coloration and decay can easily be individually controlled by a set of filters and attenuators for artistic fine tuning and abstract reverberation effects, e.g. non-exponentially decaying reverberation as in [16]. All elements of the proposed algorithm are LTI which means that the common linear signal processing theory can be applied without approximations as opposed to the time-varying algorithms, e.g. [11, 15, 17]. This kind of reverberator is suitable for many applications for example in game audio, live music, and music production.

The following section of this paper describes velvet noise. Section 3 presents the proposed algorithm. Section 4 shows how well the algorithm can fit the reverberation of a real concert hall. Section 5 compares the computational complexity and memory usage with other algorithms, finalizing this paper with a conclusion in Section 6.
Figure 1: Velvet noise generator. (a) The algorithm is initiated with an impulse, (b) which is converted into an endless pulse train, next (c) the variable delay line adds jitter to the distance between the neighboring pulses, and finally (d) the switch changes the sign of some of the pulses. The outcome of these phases are illustrated in Figure 2 (a) to (d), respectively.

2. VELVET NOISE

2.1. Generation of velvet noise

The specific kind of sparse white noise known as velvet noise was introduced in 2007 by Karjalainen and Järveläinen [11]. A more recent study [12] explains the generation of velvet noise as a jittered pulse train with randomly selected sign which gives a sequence of zeros with few noise sequence 500 samples long.

A velvet noise generator in [11] consists of a digital filter, followed by a block of three random-number generators. These three random-number generators are used to determine (i) whether each forthcoming sample in the output stream is generated, (ii) when a 1 is generated, and (iii) whether the sample is positive or negative. The output of the random-number generator is added to a delay line which delays the sample back into the random-number generator. By repeating this procedure, the output of the filter constitutes a real-time realization of velvet noise.

Let $k(n)$ be the integer sample counter and $T_d$ the integer pulse counter, $m$ the integer pulse counter, $r_1(m)$ a random value uniformly distributed in the range from 0 to 1, and $T_d$ the average distance between pulses. The $-1$ in equation (1) is to avoid coinciding pulses [12]. The sign of each pulse is chosen by the following definition:

$$s(n) = \begin{cases} 2 \text{round}[r_2(m)] - 1, & \text{if } n = k(m) \\ 0, & \text{otherwise} \end{cases}$$

where $n$ is the integer sample counter and $r_2(m)$ is another random value again uniformly distributed in the range from 0 to 1. This means that velvet noise has one parameter besides its level that is the average distance between pulses, $T_d$.

Generation of velvet noise can be described by the block diagram implementation shown in Figure 1. Figure 2 shows a signal at the four different positions, (a) to (d), of Figure 1. Both figures illustrate that (a) the algorithm is initiated by an impulse, (b) which the non-attenuating feedback loop converts into an endless pulse train, (c) the variable delay line triggered by the pulses adds jitter uniformly distributed between the pulses in the pulse train avoiding coinciding pulses, and (d) the random-controlled switch also triggered by the pulses changes the sign of some of the pulses resulting in velvet noise. The shown signal is 500 samples long with an average pulse density, $1/T_d$, of 2205 pulses/s and 44.1 kHz sample frequency corresponding to an average pulse period of 20 samples$^1$. The block diagram corresponds to reformulating (1) as

$$k(m) = \text{round}[m T_d + r_1(m) (T_d - 1)] + \text{round}[r_2(m)] (T_d - 1),$$

where $k(m)$ denotes the convolution process, $n$ the index of coefficient values which coincide with the non-zero coefficients are added together, and $d$ the random-controlled switch also triggered by the pulses changes the sign of some of the pulses resulting in velvet noise. The shown signal is 500 samples long with a lowpass filtered (fc = 1.5 kHz) velvet noise sounds smoother than Gaussian white noise presented at the same RMS level. Another conclusion from [11] was that lowpass filtered (fc = 1.5 kHz) velvet noise sounds smoother than Gaussian white noise even down to the lowest tested average pulse density of 600 pulses/s. In the more recent study [12] listening tests showed that velvet noise is the noise perceived as the smoothest among six comparable algorithms with average pulse densities at or below 2000 pulses/s further indicating that velvet noise is a well suited sparse noise for efficient late reverberation.

Two important properties of velvet noise make it very suitable for direct convolution artificial late reverberation, namely its extremely low average pulse density (meaning very few non-zero filter coefficients), which makes it computationally efficient, and its perceptible smoothness, which makes it still sound realistic. Direct convolution with velvet noise (or any other sparse sequence) is known as SFIR filtering. Velvet noise SFIR filters smears the input through time but is spectrally flat with small variations throughout the spectrum as the velvet noise. Comparable spectral variations are found for the parallel feedback comb filters in [4].

Convolution of a signal by SFIR filtering having velvet-noise coefficients can be implemented efficiently without multiplications. Most of the coefficients of the SFIR filter are zero and they must not be computed at all. Instead, the input signal is propagated in the delay line of the filter, and only those input signal samples, which coincide with the non-zero coefficients are added together to produce the output. One idea is to separately run through the indices of coefficient values $+1$ and $-1$, add the corresponding sample values taken from the delay line, and finally subtract the two sums. This convolution process can be formulated as

$$x(n) s(n) = \sum_{m_+} x[n + k(m_+)] - \sum_{m_-} x[n + k(m_-)],$$

where $x(n)$ is the input signal, $*$ denotes the convolution process.
shown in Figure 4(a) to (e), respectively.

Example signal at the five places in the algorithm, (a) to (e), is of the impulse response which in this example exactly corresponds to the length of the first three paths; the signal in Figure 4(d) continues until the $M$th path and Figure 4(e) continues even longer because of the recursive nature of the SAP filters. The length of the SFIR filters and the attenuation factor $G$ increase gradually for each path while the SFIR pulse density and the cutoff frequency of filters $H$ decrease for each path; see Figure 4(d).

The SFIR filters and their corresponding neighboring delay line can share the same memory space. The memory usage for all the SFIR filters corresponds to the length of the target impulse response. The length of each SFIR filter must be chosen shorter than the length of all the previous SFIR filters starting with zero delay for SFIR_1 and ending with a delay corresponding to the accumulated length of SFIR_1 to SFIR_{M-1} for SFIR_M. The summed signal is fed through a cascade of $K$ SAP filters to allow SFIR filters with greater sparsity to be used and to smear the transition between them.

To illustrate this, Figures 4(a) to (e) show an example impulse response at the corresponding, (a) to (e), positions of Figure 3. The signal is followed through the second path and all the way to the output. Both Figures 3 and 4 illustrate that (a) for the second path the impulse is delayed by $L_1$ samples (which corresponds to $L_1/F_1$ seconds), (b) the impulse response of SFIR_2 is a velvet noise sequence, (c) lowpass filtering and attenuation according to $H_2$ and $G_2$, (d) summation of the signal from all the paths, and (e) SAP filtered dense output signal. The figure only shows the first 300 ms of the impulse response which in this example exactly corresponds to the length of the first three paths; the signal in Figure 4(d) continues until the $M$th path and Figure 4(e) continues even longer because of the recursive nature of the SAP filters. The length of the SFIR filters and the attenuation factor $G$ increase gradually for each path while the SFIR pulse density and the cutoff frequency of filters $H$ decrease for each path; see Figure 4(d).

The SFIR filters and their corresponding neighboring delay line can share the same memory space. The memory usage for all the SFIR filters corresponds to the length of the target impulse response. The length of each SFIR filter must be chosen shorter when the frequency characteristic of the target impulse response changes more rapidly (often in the beginning). The output of each SFIR filter is filtered and attenuated corresponding to the spectral content and level of the target impulse response at the corresponding point. Lower average pulse densities are used for the SFIR filters where the coloration filter has a more lowpassed characteristic (normally towards the end of the impulse response).

The $N$th order SAP filter is given by the transfer function [18]

$$A(z) = \frac{g + z^{-N}}{1 + g z^{-N}},$$

where $g$ is the allpass coefficient which for all the SAP filters is 0.618, derived from the golden ratio, to maximally reduce the peak power of impulses [19, 20]. The order of the SAP filters are distributed as integer values between 1 (first order) and the overall average pulse period $T$. The number of cascaded SAP filters and the overall average pulse period is calculated as the mean of all SFIR average pulse periods.
their orders are selected so that they together fill out the gaps of the SFIR filters; meaning that increasing sparsity of SFIR filters demands more SAP filters to make up for the missing SFIR pulses. A sufficient number of SAP filters and their orders are selected so that the probability distribution of a SAP filtered velocity noise signal with the overall average pulse density approximates a Gaussian distribution. An SFIR average pulse density starting at 100 pulses/s and a cascade of seven SAP filters has been found to be a good compromise for a diffuse concert hall; this configuration will be used in the reverberator example presented in the next section.

The algorithm produces late reverberation. The early reflections can be approximated in many ways, e.g. as described in [15, 21, 22]. A simple example of a complete reverberator containing direct sound, early reflections, and late reverberation is shown in Figure 5. The early reflections are reproduced by convolution (FIRr) of the first part of the impulse response excluding the direct sound impulse. The input signal is delayed, $z^{-D}$, and passed through the late reverberation algorithm before mixing with the direct sound and early reflections. The mixing can include additional filtering, delay etc. of the signals; in the complete reverberator used in the next Section the mixing procedure is just an addition of the three signals. FIRr and $z^{-D}$ can share the same delay-line memory.

The impulse response of a cascade of several SAP filters is more sparse in the beginning and grows more dense as the amplitude decays and the peak power is delayed, as seen in Figure 4(e). To camouflage this sparsity and to make up for the delay, $z^{-D}$ is less than the length of FIRr; more exactly FIRr minus the amount of samples as the sum of all the SAP filter orders. Hereby the sparse part of the SAP filter impulse response occurs while the early reflections still sounds. The SAP filters of course only smear the signal forward in time, making the output of each path overlap. The first path is lacking an overlap which is compensated for by a slight amplification of $G_1$, depending on the sum of the order of the SAP filters.

### 4. MODELING EXAMPLE

#### 4.1. Measured impulse response

As an example of how well the reverberator can replicate real world impulse responses we have used a good quality well documented impulse response recorded from the concert hall in the Finnish city Pori as our target impulse response [23]. The chosen impulse response was recorded with the source on the stage, receiver at the floor far from the stage, and with omni-directional microphone. The impulse response is named `s1_r3_o.wav` in

![Figure 5: Complete reverberator containing direct sound, early reflections, and late reverberation; and mixing of the three signals. The content of the block named “Reverb” is shown in Figure 3.](image)

Figure 5: Complete reverberator containing direct sound, early reflections, and late reverberation; and mixing of the three signals. The content of the block named “Reverb” is shown in Figure 3.

The content of the block named “Reverb” is shown in Figure 3.

![Figure 6: (a) Time signal and (b) spectrogram of measured impulse response from the concert hall in Pori, Finland [23].](image)

Figure 6: (a) Time signal and (b) spectrogram of measured impulse response from the concert hall in Pori, Finland [23].

![Figure 7: (a) Gradually increasing length of the 20 frames starting from 100 milliseconds. (b) Decreasing pulse density with increasing filter number.](image)

Figure 7: (a) Gradually increasing length of the 20 frames starting from 100 milliseconds. (b) Decreasing pulse density with increasing filter number.

[23]: the same impulse response has been used to test the performance of the reverberator described in [11]. Figure 6 shows the time signal and spectrogram of the impulse response. The sample frequency of the measured and replicated impulse response is 44.1 kHz.

### 4.2. Replicated impulse response

We consider the late reverberation of the impulse response from Pori to span from 100 milliseconds to when the impulse response reaches zero amplitude after approximately 2.1 seconds\(^3\). Filters are fitted by linear predictive coding (LPC) to $M$ non-overlapping rectangular windows of the measured late reverberation. The length of the analyzed windows corresponds to the length of the SFIR filters and thereby selects the input to each path. The number and length of windows was chosen based on inspection of the spectrogram in Figure 6(b). In the beginning of the signal the rapid change of frequency characteristics motivated for shorter windows, while

\(^3\)The signal reaches zero amplitude (no noise-floor) because it has been de-noised, see [23] for details. For a non-denosed impulse response the signal should be faded out when all frequency regions have reached the noise floor to avoid convolution with the noise floor.
at the end larger window lengths are sufficient. Figure 7(a) illustrates the gradually increasing window length for higher filter numbers resulting in \( M = 20 \) LPC filters and paths\(^4\). The LPC filters gave a good fit down to order 10 which was chosen as the LPC filter order. Figure 8 shows the magnitude response for a selection of the fitted LPC filters including attenuation; the tendency is clearly a gradually increasing lowpass characteristic and increasing attenuation for higher filter numbers. The increasing lowpass characteristic was exploited by lowering the average pulse density of the higher filter numbers as shown in Figure 7(b); the average pulse density was linearly distributed from 100 to 40 pulses/s for the 20 filters.

The number of cascaded SAP filters was chosen to be \( K = 7 \) by the procedure described in Section 3 (approximating a Gaussian amplitude distribution). The order ranges from 1 to the overall mean of the average pulse period, \( T_{\text{avg}} = 630 \) samples. The exact order of the SAP filters are 1, 64, 140, 209, 442, 555, and 630. The attenuation for the paths, \( G_1 \) to \( G_{20} \), was calculated as the average power of a one-second velvet noise sequence with the same average pulse density as the SFIR in that path filtered by the LPC filter and the cascade of SAP filters. The lack of smearing in the first path is compensated for by an amplification; \(+3\) dB has shown to be sufficient for the chosen SAP filters.

The direct sound and early reflections are generated by convolution of the first 100 milliseconds of the original impulse response. The time signal and spectrogram of the replicated impulse response is shown in Figure 9, which closely resembles Figure 6. Figure 10 shows a comparison of the reverberation times of the measured and the generated impulse response from Figure 6 and 9, respectively. The reverberation times are calculated as \( T_{30} \). The generated impulse response varies in any octave band no more than \( \pm 7\% \) relative to the measured impulse response.

Informal listening tests revealed that the perceived properties of the impulse response and artificial reverberation of sounds generated from the algorithm were very similar to the original. The generated impulse response and sound examples are available on the Internet\(^5\). The sonic performance of the algorithm can easily be evaluated for other sounds by convolution with the impulse response because the algorithm is linear and time-invariant.

\^4\text{The border sample indexes of the 20 windows are as follows: 4411, 5672, 7214, 9044, 11171, 13602, 16343, 19400, 22779, 26484, 30521, 34895, 39609, 44669, 50077, 55837, 61954, 68431, 75271, 82477 and 90053.}

\^5\text{http://www.acoustics.hut.fi/go/dafxl3-v screverb/}

4.3. Abstract reverberation effect

The settings for the generated impulse response shown in Figure 9 has been altered to illustrate how well the algorithm also works for producing abstract reverberation effects. In this example only the 20 attenuators, \( G_1 \) to \( G_{20} \), have been manipulated\(^6\) to give an impulse response that has an increasing amplitude. Figure 11 shows the resulting time signal and spectrogram. The shown abstract reverberation effect sounds somewhat similar to a compressed and gated reverb. Many other kinds of abstract effects can easily be made with this algorithm including time-reversed RIR and other non-decaying or pulsating impulse responses.

5. IMPLEMENTATION COST COMPARISON

The implementation cost of the algorithm is compared to direct convolution and partitioned fast convolution. Table 1 shows the

\(^6\text{The exact } G\text{-values used for the abstract reverb effect are relative to the values found in Subsection 4.2. The relative values increases } +3.47 \text{ dB for each path starting with 0 dB for } G_1.\)
number of floating point operations (FLOPs) per sample and the number of signal memory samples required for the three reverberation algorithms for a 2-second long impulse response, as in the previous section. The listed values for the direct convolution are based on the direct form implementation. The values for the partitioned fast convolution are taken from the recent improvement of the algorithm by Wefers and Vorländer [10, Table 1]. The settings for replicating the impulse response in Section 4 was used for the computational cost and memory usage calculations listed for the proposed algorithm. The implementation cost of the early reflections is not included in the calculations. Table 1 shows that the proposed algorithm is approximately as efficient computationally as the partitioned convolution and over 100 times more efficient than the direct convolution. The memory consumption of the new algorithm is approximately as efficient computationally as the partitioned convolution and approximately 50% smaller than that of the partitioned convolution algorithm.

Table 2 specifies what kind of operations (multiplication or addition) and from which part of the algorithm the 527 FLOPs originate. \( \oplus \) denotes the summation before the SAP filters. Note that the SFIR filter uses zero multiplications and that approximately 73% of the operations (400 FLOPs) are spend on the spectral coloration filterbank. The computational complexity listed is for FLOPs only, ignoring any branching operations, specialized signal processing instructions etc. The memory usage listed is for the signal memory only, ignoring any memory needed for filter coefficients.

6. CONCLUSION

This paper presented a novel algorithm for simulating the late part of room reverberation. The idea is to use a bank of sparse FIR (SFIR) filters followed by spectral coloration filters, from which the summed output is fed through a cascade of several Schroeder allpass (SAP) filters. The coefficients for the SFIR filters are obtained from velvet noise sequences, which are proven to provide smooth responses even with very low pulse densities (down to 0.1-0.2% non-zero elements). Moreover, the sparsity of the SFIR filters varies along the reverberation tail with sparser filters towards the end where the impulse response has a more lowpassed characteristic. The coloration filters can be designed by fitting LPC filters to partitioned variable-length segments of a target impulse response in order to obtain realistic coloration for the reverb, and the SAP filters are used to smooth out the transition between SFIR filters. The inclusion of SAP filters allows for using even sparser SFIR filters; this principle can also be used in similar algorithms to lower the computational cost considerably, e.g. [13, 14, 11, 15, 17].

The performance of the proposed algorithm was demonstrated with a modeling example, and the results showed that the algorithm is able to model the overall characteristics of the target concert hall impulse response. The design procedure allows a flexible parametric approximation of the target late part of the impulse response, and additionally, the proposed reverb is computationally efficient providing a clear advantage over the direct convolution and comparable to the FFT-based partitioned convolution method, but with nearly half the memory usage. Sound examples are available online at http://www.acoustics.hut.fi/go/dafx13-vscreverb/.

7. ACKNOWLEDGMENTS

This work was conducted when Bo Holm-Rasmussen was a visiting student at Aalto University, Espoo, Finland. The work of Heidi-Maria Lehtonen was supported by the Finnish Cultural Foundation.

8. REFERENCES


