A VST REVERBERATION EFFECT PLUGIN BASED ON SYNTHETIC ROOM IMPULSE RESPONSES

Christian Borß
Institute of Communication Acoustics,
Ruhr-Universität Bochum
Bochum, Germany
christian.borss@ruhr-uni-bochum.de

ABSTRACT

In this paper we present a newly developed VST reverberation effect plugin ("HybridReverb") based on synthetic room impulse responses (RIRs). We detail how we choose proper parameters for the synthesis of RIRs as presets for our convolution-based reverberation effect. The implemented stereo/surround plugin provides natural sounding reverberation based on physical principles. The newly developed convolution engine features signal processing with low latency and uniform processing load.

1. INTRODUCTION

The history of digital filters for the generation of artificial reverberation goes back to the early sixties when Schroeder used feedback loops with delay lines to create reverberation effects [1, 2]. The first commercial product which provided reverberation effects based on digital delay lines and which could actually compete with the sound quality of the analog reverberation effects at that time was the EMT-250 developed by Blesser in 1976 [3]. Moorer then added IIR filters in the feedback loop of the comb filters to model frequency dependent absorption [4]. Stautner and Puckette proposed a feedback-delay network as generalization for artificial reverberators based on digital delay networks [5]. An overview over these filter networks and design guidelines can be found, e.g., in [6].

Filter networks based on feedback loops with delay lines have great appeal due to their low computational complexity, but it's not an easy task to yield truly colorless reverberation with these filters. As stated by Gardner who used nested allpass filters for creating reverberation [7], such filter networks were used more for practical reasons in the past, because digital signal processors and personal computers were not capable of convolving an audio signal with a sufficiently long RIR at the time the filter networks were proposed. A convolution with a measured or synthesized RIR can avoid the coloration problems. Rendering a synthetic RIR provides some degree of freedom which can be used, e.g., for artistic design [8]. Moorer suggested a random noise signal attenuated by a frequency dependent exponential decay as synthetic RIR [4]. An analysis of such a reverberator can be found, e.g., in [9]. Examples for the synthesis of RIRs can be found, e.g., in [10, 11].

All above mentioned reverberators are suited for the implementation of a perceptual approach for the generation of synthetic acoustics [12, 13, 14]. An accurate reproduction or prediction of room acoustics can be achieved by physically-based auralization approaches [15, 16, 17, 18, 19]. An overview over this field of research can be found, e.g., in [20, 21].

Physical accuracy is actually not necessary for a reverberation effect. However, using physical principles for the implementation of a perceptual approach allows for highly plausible and very natural-sounding results. In this paper we use such an approach to synthesize natural sounding reverberation using acoustical design parameters such as the room size, the frequency dependent reverberation time, and the echo density profile. The following Sec. 2 details the generation of RIRs and the parameters which we use to synthesize a set of RIRs as presets for a newly developed reverberation effect. In Sec. 3 we discuss signal processing aspects of the implemented stereo/surround room effect.

Audio examples, the implemented software, the source code, and the presets discussed in Sec. 2.1 are available at [22].

2. RENDERING MODEL

The aim of an auralization software is to create the sound field of a simulated environment at a given listener position. In general, a sound field can be reproduced by using a set of microphones with a specific directivity pattern [23] and by feeding a matching set of speakers with the recorded signals. By measuring RIRs for discrete sound source positions and convolving a mono signal with these impulse responses, it is possible to auralize sound sources at the chosen positions. Instead of measuring RIRs in a physical environment, i.e., in an existing room, we synthesize room impulse responses. These synthetic RIRs can be regarded as impulse responses "measured" in a virtual environment with a virtual sur-

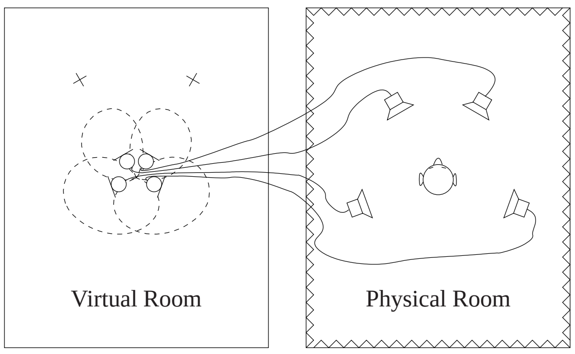

Figure 1: *Generation of a sound field as simulated in a virtual environment at a given listener position (dashed lines: directivity pattern of the virtual coincidence microphone array; crosses: discrete sound source positions)*

round sound microphone array. Figure 1 illustrates this concept. For a stereo reverberation effect, we generate impulse responses for two sound source positions which are marked by the crosses in Fig. 1. The dashed lines indicate the directivity pattern of the virtual microphone array which results from the panning algorithm that is used to render sound sources between the discrete loudspeaker positions (this relation is discussed, e.g., in [24]).

For the generation of synthetic room impulse responses, we use the auralization software "tinyAVE" [8]. This software was initially developed to create virtual environments for speech communication, but it is also capable of plausibly simulating the acoustics of a concert hall [25]. The following section gives a brief introduction into the rendering model in order to explain the meaning of the parameters given in Sec. 2.1. A more detailed description of the signal generation architecture can be found in [8] and [26].

"tinyAVE" uses a modified image source model [27] and an improved statistical time-frequency model [4] to synthesize acoustics according to a given set of room acoustic parameters, i.e., the room geometry, the frequency dependent reverberation time, and the echo density profile[1]. The image source model is used to simulate the direct sound and the specular reflections up to a certain maximum reflection order. With vector base intensity panning [28], a variant of vector base amplitude panning [29] based on Gerzon's "energy vector" [30], we yield proper "phantom sources" (auditory events) between our speakers.

For a plausible result, it is crucial to incorporate specular reflections up to a minimum order. Due to its algorithmic simplicity, we use a shoebox geometry for the simulation of specular reflections. A drawback of this geometry is the resulting coloration that is due to the comb filter effect as a result of the regular structure of the mirror image sources. By randomizing the positions of the higher-order image sources, we can avoid the coloration while retaining the increasing density of reflections which can be observed in a physical environment. This randomization is illustrated in Fig. 2.

For the late reverberation, we use an extended version of the statistical time-frequency model proposed by Moorer [4] which models both, time and frequency behavior, of reverberation tails[2].

[1]the different models are one of the reasons for the name "*Hybrid*Reverb"

[2]the signal processing in time and frequency domain is another reason for the name "*Hybrid*Reverb"

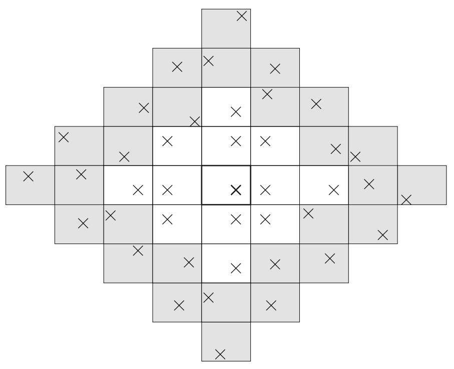

Figure 2: *Image sources of a sound source in a shoebox room; we randomize the positions of higher-order image sources (shaded area) to avoid coloration*

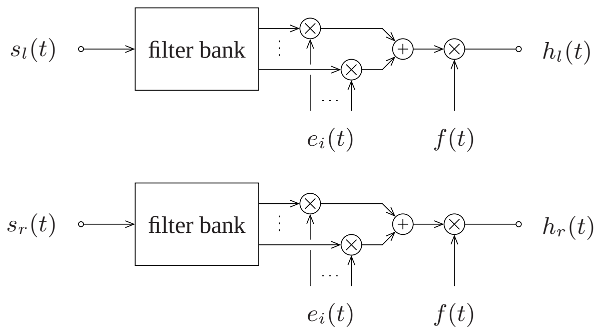

Figure 3: Generation of the late reverberation tails $h_l(t)$ and $h_r(t)$ for the left and the right channel from uncorrelated white noise $s_l(t)$ and $s_r(t)$; the frequency dependent reverberation time is adjusted by the exponential decay $e_i(t)$ in frequency band i; the fade-in function $f(t)$ is used for a smooth combination with the early reflection model

In this approach, a random noise signal attenuated by a frequency dependent exponential decay is used for the synthesis of an artificial late reverberation tail as shown in Fig. 3. Our late reverberation rendering model takes also the echo density profile [31, 32] into account for a discrete noise signal by thinning out the Gaussian noise signal $g(n)$ to a desired density $p(n)$,

$$s(n) = \begin{cases} g(n) & u(n) \leq p(n)/f_s \\ 0 & \text{else} \end{cases}, \tag{1}$$

where $s(n)$ is a sparse version of the noise signal at the sampling frequency f_s using an additional random signal $u(n)$ which is uniformly distributed between 0 and 1.

In order to smoothly combine the output of the image source model and the output of the late reverberation rendering model, we use a fade-in function $f(t)$ for the late reverberation tail as shown in Fig. 3. The method we use to derive the fade-in function directly from the parameters of the image source model is presented in [26].

The proposed convolution-based statistical late reverberation model is superior to the feedback networks mentioned in Sec. 1 for three reasons: it allows to combine the late reverberation model and the image source model by an optimally matching fade-in function, it provides a greater influence on the modeling of the late reverberation tail (e.g., by a given echo density profile), and it has a greater robustness against coloration.

2.1. Preset Parameters

In our approach, we use the room geometry, the frequency dependent reverberation time, and the echo density profile as rendering parameters. The direct sound as well as the early specular reflections are rendered by simulating reflections in a shoebox room. For the presets of our room effect plugin, we choose the length L_x, the width L_y, and the height L_z of the room as follows:

$$L_x = 1.9L_z \tag{2}$$

$$L_y = 1.4L_z \tag{3}$$

This ratio is often used for the construction of professional sound studios due to the favorable distribution of room resonances. Table 1 lists the geometries we have chosen for our presets.

label	L_x / m	L_y / m	L_z / m	T_{60} / s
“bathroom”	4.65	3.43	2.45	0.79
“livingroom”	6.65	4.90	3.50	1.00
“studio”	9.31	6.86	4.90	1.19
“small concert hall”	13.30	9.80	7.00	1.40
“large concert hall”	26.60	19.60	14.00	1.81
“huge concert hall”	53.20	39.20	28.00	2.21

Table 1: *Chosen room dimensions and reverberation times recommended by [33] for music reproduction in rooms of this size*

We choose the direction of the sound sources in the virtual room at $\pm 30^\circ$ relative to the viewing direction of the listener. This is identical to the directions of the front speakers in a 5.1 speaker setup as recommended by ITU-R BS.775-1 [34]. As a result, the stereo image of the input signal of our reverberation effect plugin is preserved, because the direct sound is not panned between the speakers for this configuration. Because the stage of a concert hall like the *Wiener Musikvereinssaal* is located at the head side of the shoebox, we place the listener and the two sound sources in our virtual room so that the center of the spanned triangle is at $(L_x - L_y\sqrt{3}/6 - 0.5, L_y/2, 1.7)^T$. By this, there is no lateral preference, we can assure a minimum distance of 0.5 m between the sound sources and the back wall, and the listener as well as the sound sources are approximately at the height of the ears of a standing person. Figure 4 shows the resulting configuration for a source distance of 1m in the “bathroom” preset. Furthermore, we attenuate the first reflection on the floor by 9 dB to reduce the comb filter effect for large source distances where the direct sound and the reflection on the floor are close together in time domain.

The German industrial norm *DIN 18041* is a good basis for the choice of the reverberation times in our presets, as it gives recommendations for reverberation times for music reproduction, speech, and education in small to medium sized rooms. In a first attempt, we used a frequency dependent reverberation time which is given by the average of the upper and the lower limits specified by the norm. Informal listening tests using dry recordings of speech (the author’s voice, recorded with an *AKG C 414 B-ULS* microphone in an anechoic chamber) and classical music (W. A. Mozart, an aria of *Donna Elvira* from the opera *Don Giovanni* [35]) in a selected room configuration (26.2 x 19.3 x 13.8 m^3, $T_{60} = 1.8$ s) showed that the resulting reverberation was perceived as sounding too crisp. As a consequence, we choose a frequency dependent reverberation time with a steeper slope for the high frequencies as shown in Fig. 5. Multiplied with the nominal values listed in Tab. 1, this yields the final frequency dependent reverberation times from which the reflection filters of the image source model as well as the late reverberation tail are constructed.

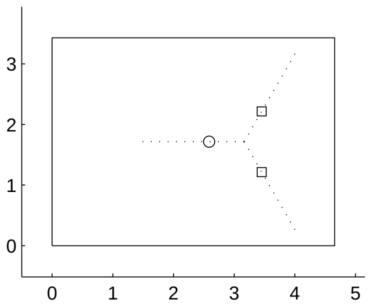

Figure 4: *Listener (circle) and source (squares) positions [m] in the “bathroom” preset*

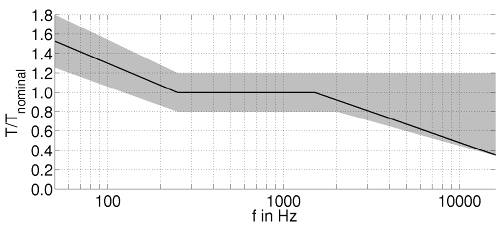

Figure 5: *Chosen frequency dependent reverberation time (solid line) and the range recommended by [33] for music reproduction (shaded area) normalized to the nominal reverberation time as listed in Tab. 1*

The above mentioned informal listening test also showed, that a constant echo density profile of 5000 impulses per second yields a late reverberation that is sufficiently diffuse while being not too “dense”. We also tested an echo density profile with a quadratically increasing density identical to the increasing density of the image source model but limited to 5000 impulses per second. We found that the two versions were not distinguishable. A possible explanation for this result is that the late reverberation is already sufficiently dense at the point when it is faded in.

3. IMPLEMENTATION AS VST PLUGIN

A natural sounding multi-speaker reverberation effect for a stereo input signal can be created by modelling the stereo sound source by two point sources in a virtual environment and by auralizing this environment for a given multi-speaker setup as depicted in Fig. 1. In order to take into account the transfer function from each virtual point source to each virtual microphone, a fully meshed filter network with a filter between each input and each output channel is required. For practical reasons, we restrict the number of used speakers to the front and the rear speakers of an ITU-R BS.775-1 compliant speaker setup [34]. By splitting up the fully meshed filter network into two filter networks which feed the front and the rear speakers, respectively, we obtain two fully meshed stereo filters. This allows for creating a surround sound reverberation effect for an audio signal processing framework which only supports stereo effects, i.e., filters with two input and two output channels. These fully meshed stereo filter networks can then be implemented by four finite impulse response (FIR) filters as shown in Fig. 6 where the output signals $y_l(t)$ and $y_r(t)$ result from the convolution of the input signals $x_l(t)$ and $x_r(t)$ with the impulse responses $h_{l,l}(t)$, $h_{l,r}(t)$, $h_{r,l}(t)$, and $h_{r,r}(t)$.

We have implemented such a stereo/surround reverberation effect using the *Virtual Studio Technology* (VST) framework developed by Steinberg [36] and *FFTW*, a library for the fast Fourier transformation [37]. The implemented VST room effect which uses a set of impulse responses as presets was developed for real-time applications like software-based musical instruments.

An effect plugin for musical instruments requires signal processing with low latency, e.g., by means of a segmented convo-

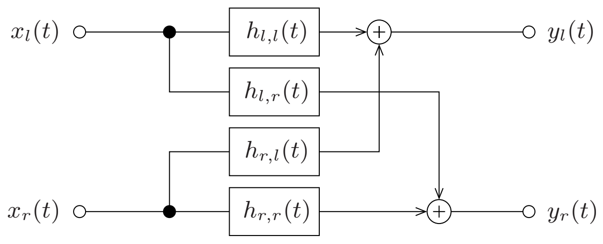

Figure 6: *Fully meshed stereo filter*

lution [38, 39] where the segment length can be chosen to yield a desired compromise between latency and computational complexity. By partitioning the impulse response into segments of increasing length, the computational complexity can be further improved [40]. A method for the determination of a theoretical optimal segmentation can be found in [41]. In practice, however, this does not necessarily result in an optimal performance due to hardware effects like CPU cache miss [42, 43].

The hybrid method proposed by Gardner [44] uses the direct form of the convolution formula for the first part of the impulse response and a segmented convolution with non-uniform segment length for the rest. This allows for an efficient implementation with "zero" input-output latency. In our case, a convolution with zero latency is not essential for two reasons. The first reason is that a latency below a certain threshold is not noticeable (about 2-6 ms for musicians [45]). The second reason is that there is actually no need for sample-wise processing, if the audio hardware processes audio buffers of fixed block sizes anyway.

Even though a segmented convolution with non-uniform segment lengths may result in the lowest possible computational complexity, it is not necessarily the best solution for a realtime implementation. The reason for this is the non-uniform processing load for the different segments. As the realtime behavior of a signal processing block is defined by the worst case, i.e., the longest processing time for the individual segments, it is desireable to have a segmented convolution with uniform processing load. In our implemented system, we use a segmented convolution with uniform processing load using three segment lengths S, M, and L as shown in Fig 7. In the following Sec. 3.1 we explain how a uniform processing load can be achieved in general with two segment lengths. In Sec. 3.2 we subsequently discuss how we achieve uniform processing load with three segment lengths using a novel load shaping mechanism.

3.1. Segmented Convolution with Uniform Processing Load

The processing latency of a segmented convolution is given by the segment length. A short segment length will result in a short processing latency at the cost of computational complexity. By splitting up the impulse response into an initial part $h_m(t)$ with segments of the length M and a final part $h_l(t)$ with segments of the

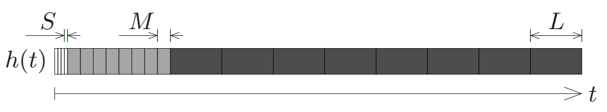

Figure 7: *Segmentation of the impulse response $h(t)$ for a convolution with three segment lengths*

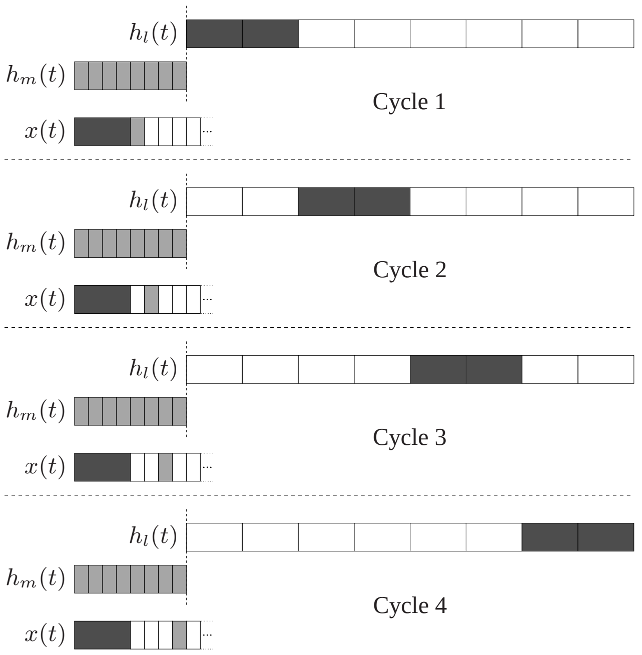

Figure 8: *Segmented convolution of an input signal $x(t)$ with an impulse response split into two parts $h_m(t)$ and $h_l(t)$ with different segment lengths M and L; $L/M = 4$ processing cycles result in an uniform processing load; the light gray blocks mark the blocks which are processed by the segmented convolution with $h_m(t)$ and the dark gray blocks mark the blocks which are processed by the segmented convolution with $h_l(t)$*

length L where $L > M$, we can reduce the computational complexity compared to a convolution with uniform segment lengths M [40]. In this approach, the implementation of the convolution is split into two segmented convolutions with uniform segment lengths. In a straightforward implementation, the shorter segments are processed in every processing step and the longer segments are processed whenever the input buffer filled with the shorter segments is filled up to the full length of the long segments. As a result, the computational load is increased every L/M processing step by the amount for the processing of the long segments.

Instead of processing all long segments every L/M processing steps, we can split up the work load into L/M equal parts which are processed at different cycles as shown in Fig. 8. In order to do so, we need $2L/M$ segments of the length M. As the length of $h_l(t)$ is not necessarily an even multiple of L^2/M, the workload for the different processing cycles will only be approximately equal in practice. In addition, the Fourier transform and its inverse have to be applied in the first and the last processing cycle. Because of this, it is advisable to process a lower amount of segments in the first and the last processing cycle.

3.2. Load Shaping Mechanism

The fact that we are processing more than just one FIR filter for a fully meshed stereo filter as shown in Fig. 6 can be exploited to further reduce the computational complexity by a third segment length while preserving the property of uniform processing load. We implement this by splitting up the impulse response into a first

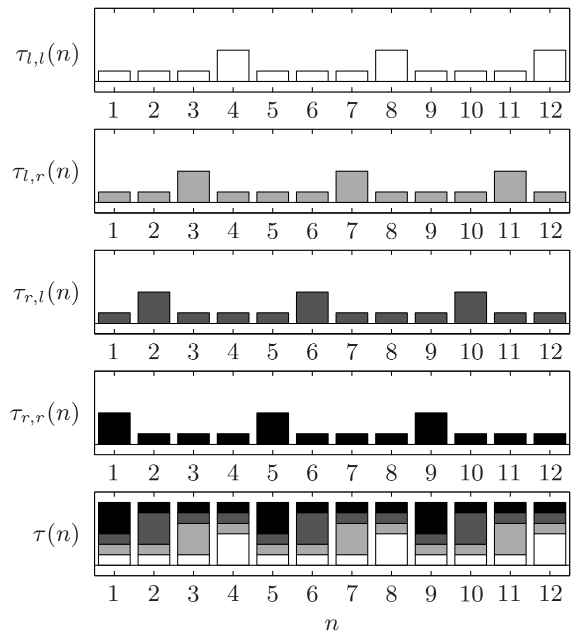

Figure 9: *Qualitative evolution of the processing time $\tau_{i,j}(n)$ at processing step n for the 4 FIR filters shown in Fig. 6 and the resulting processing time $\tau(n)$ for all filters*

part $h_s(t)$ with segments of the length S, a second part $h_m(t)$ with segments of the length M where $M > S$, and a third part $h_l(t)$ with segments of the length L where $L > M$. The shortest segments are processed in every processing step whereas the other segments are processed according to Sec. 3.1 whenever a buffer filled with the input segments contains at least M samples. For a single filter, this results in a non-uniform processing load as stated in Sec. 3. By pre-processing a number of segments with zero values and discarding the output[3], we can shift the maximum of the processing load. Hence, if we choose $M = 4S$ and initialize in this way the four FIR filters differently, we yield a uniform overall processing load for three segment lengths as shown in Fig. 9.

3.3. Performance

The shortest segment length, S, can be selected arbitrarily to yield a given processing latency. The length of the medium segments, M, should be set to $4S$ to yield a uniform processing load for a fully meshed stereo filter. The length of the longest segments, L, can be freely chosen and tuned to optimize the computational load. We have implemented a simple benchmark program for our convolution library ("libHybridConv") which estimates the CPU load for segmented convolutions with different combinations of S, M, and L. The estimated CPU load is derived from the processing time for an audio signal of a given length. Table 2 lists the estimated CPU load measured on a Debian GNU/Linux PC with a 1.86 GHz Intel Core2 Duo CPU and 3 GB RAM for a single FIR filter with 86901 taps at a sampling frequency $f_s = 48$ kHz. Please note that the symmetric multiprocessing (SMP) feature of the CPU was intentionally not used in order to gain a better comparability with single-core systems. Modern multi-core processors provide for parallel computing by distributing the work load among the processor cores. Our implemented plugin uses OpenMP [46] to speed up the computation of the outputs of the FIR filters on multi-core or multi-processor systems. OpenMP is an extension to the programming languages C/C++/Fortran which supports loop level parallelization on shared-memory systems. In our implementation, the four FIR filters of each fully meshed stereo filter are processed in parallel. This allows to utilize up to four CPU cores of a state-of-the-art multi-core system. If the VST host software also uses loop level parallelization for the execution of the different stereo filters, the processing load for our fully meshed surround filter network can be distributed even over eight CPU cores. Furthermore, we use *Single Instruction, Multiple Data* (SIMD) instructions to multiply, add, or subtract four single precision floating point variables with a single instruction using the *Streaming SIMD Extensions* (SSE) of current x86 processors.

S / samples	M / samples	L / samples	CPU load
32	128	2048	5.40 %
64	256	4096	2.83 %
128	512	4096	2.49 %
256	1024	4096	2.31 %
512	2048	8192	2.25 %

Table 2: *Load of one CPU core for a segmented convolution with a single FIR filter with 86901 taps measured on a PC with a 1.86 GHz Intel Core2 Duo CPU and 3 GB RAM; only one CPU core was used for a better comparability with single-core systems*

To be able to make a rough estimate of the runtime behavior of a fully meshed stereo filter in a "real world" application, we used the implemented software as a VST room effect for a commercial VST instrument[4] and a commercial VST host[5]. On the same PC, running Windows XP Professional, we observed a CPU load of approximately 7 % for a fully meshed stereo filter with 4 FIR filters of the same length and a segmentation with $S = 64$, $M = 256$, and $L = 4096$ samples (the values varied mainly between $5 \ldots 9$ %). The given value includes the load for the VST host and the load for the VST instrument which was idle during the measurement. The observed load is in good accordance with the estimated value, as the CPU load reported by the Windows Task Manager is given as a mean value of both CPU cores ($7 \geq 2.83 \cdot 4/2$). As an effect plugin for a VST instrument requires signal processing with low latency and low computational complexity, we can conclude that the implemented software is well suited for an application as a room effect plugin for a VST instrument.

4. SUMMARY AND CONCLUSIONS

In this paper we adopted a plausible auralization approach to synthesize room impulse responses based on room acoustic parameters such as the room size, the frequency dependent reverberation time, and the echo density profile. We showed how this can be used to create a set of natural sounding room impulse responses as

[3]in practice we only initialize the step counter of our implemented filter with a different value

[4]Best Service *Galaxy 2* software piano [47]

[5]Steinberg *V-STack* [36]

presets for a newly developed convolution-based VST reverberation effect plugin.

We discussed signal processing aspects related to the implementation of the convolution and described how we yield a uniform processing load for a segmented convolution with three segment lengths.

Furthermore, we measured the performance of the implemented software and found that our convolution engine is well suited for the usage as a room effect plugin for a software-based musical instrument.

5. REFERENCES

[1] M. R. Schroeder and B. F. Logan, “”Colorless” Artificial Reverberation,” *Journal of the Audio Engineering Society*, vol. 9, no. 3, pp. 192–197, 1961.

[2] M. R. Schroeder, “Natural Sounding Artificial Reverberation,” *Journal of the Audio Engineering Society*, vol. 10, no. 3, pp. 219–223, 1962.

[3] B. Blesser et al., “Electric Reverberation Apparatus,” Jan. 1980, United States Patent 4,181,820.

[4] J. Moorer, “About this Reverberation Business,” *Computer Music Journal*, vol. 3, no. 2, pp. 13–18, 1979.

[5] J. Stautner and M. Puckette, “Designing Multi-Channel Reverberators,” *Computer Music J.*, vol. 6, no. 1, pp. 52–65, 1982.

[6] J.-M. Jot, “Digital delay networks for designing artificial reverberators,” in *90th AES Convention*, Paris, France, 1991.

[7] W. G. Gardner, “A Realtime Multichannel Room Simulator,” in *124th Meeting of the Acoustical Society of America*, New Orleans, LA, USA, Nov. 1992.

[8] C. Borß and R. Martin, “An Improved Parametric Model for Perception-Based Design of Virtual Acoustics,” in *AES 35th Int. Conference*, London, UK, Feb. 2009.

[9] J.-M. Jot and L. Cerveau and O. Warusfel, “Analysis and Synthesis of Room Reverberation Based on a Statistical Time-Frequency Model,” in *103rd AES Convention*, New York, NY, USA, 1997.

[10] J.-P. Vian and J. Martin, “Binaural Room Acoustics Simulation: Practical Uses and Applications,” *Applied Acoustics, Special Issue “Auditory Virtual Environment and Telepresence”*, vol. 36, no. 3&4, pp. 293–305, 1992.

[11] D. Murphy et al., “Hybrid Room Impulse Response Synthesis in Digital Waveguide Mesh Based Room Acoustics Simulation,” in *11th Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finnland, 2008.

[12] J.-M. Jot and O. Warusfel, “A Real-Time Spatial Sound Processor for Music and Virtual Reality Applications,” in *International Computer Music Conference*, Banff, Canada, 1995.

[13] R. Väänänen et al., “Efficient and Parametric Reverberator for Room Acoustics Modeling,” in *International Computer Music Conference (ICMC ’97)*, Thessaloniki, Greece, Sep. 1997.

[14] A. Silzle, P. Novo, and H. Strauss, “IKA-SIM: A System to Generate Auditory Virtual Environments,” in *116th AES Convention*, Berlin, Germany, 2004.

[15] G. M. Naylor, “ODEON - another hybrid room acoustical model,” *Applied Acoustics*, vol. 38, no. 2-4, pp. 131–143, 1993.

[16] W. Ahnert and R. Feistel, “EARS Auralization Software by Ahnert,” *J. Audio Eng. Soc.*, vol. 41, no. 11, pp. 894–904, 1993.

[17] B.-I. L. Dalenbäck, “Verification of Prediction Based on Randomized Tail-Corrected Cone-Tracing and Array Modeling,” *J. Acoust. Soc. Am.*, vol. 105, no. 2, pp. 1173, Feb. 1999.

[18] T. Funkhouser et al., “A Beam Tracing Method for Interactive Architectural Acoustics,” *J. Acoust. Soc. Am.*, vol. 115, no. 2, pp. 739–756, Feb. 2004.

[19] T. Lentz and D. Schröder and M. Vorländer and I. Assenmacher, “Virtual Reality System with Integrated Sound Field Simulation and Reproduction,” *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 2007, Article ID 70540.

[20] M. Kleiner and B.-I. Dalenbäck and U.P. Svensson, “Auralization - An Overview,” *JAES Journal of the Audio Engineering Society*, vol. 41, no. 11, pp. 861–875, Nov. 1993.

[21] M. Vorländer, *Auralization: Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality*, Springer, Berlin, Germany, first edition, 2007.

[22] C. Borß, “HybridReverb reverberation plugin and 51 presets with impulse responses for front and rear speakers,” 2009, URL: `http://www2.ika.rub.de/HybridReverb/`.

[23] A. Laborie and R. Bruno and S. Montaya, “A New Comprehensive Approach of Surround Sound Recording,” in *114th AES Convention*, Amsterdam, The Netherlands, Mar. 2003.

[24] AES Staff Writer, “Novel Surround Sound Microphone and Panning Techniques: A Digest of Selected AES Convention and Conference Papers,” *J. Audio Eng. Soc.*, vol. 52, no. 1/2, pp. 74–80, 2004.

[25] C. Borß, “Audio examples rendered with tinyAVE,” 2009, URL: `http://www2.ika.rub.de/tinyAVE_orchestra/`.

[26] C. Borß, “A Novel Approach for Optimally Matching a Late Reverberation Model to an Image Source Model - Or: What Does a Football Have to Do With Shoebox Shaped Rooms?,” in *Proc. of the EAA Symposium on Auralization*, Espoo, Finland, Jun. 2009.

[27] J.B. Allen and D.A. Berkley, “Image method for efficiently simulating small-room acoustics,” *J. Acoust. Soc. Am.*, vol. 65, no. 4, pp. 943–950, 1979.

[28] J.-M. Pernaux and P. Boussard and J.-M. Jot, “Virtual sound source positioning and mixing in 5.1 implementation on the real-time system genesis,” in *First COST-G6 Workshopon Digital Audio Effects (DAFx-98)*, Barcelona, Spain, 1998.

[29] V. Pulkki, “Localization of Amplitude-Panned Virtual Sources - Part 2: Two- and Three-Dimensional Panning,” *J. Audio Eng. Soc.*, vol. 49, no. 9, pp. 753–767, 2001.

[30] M. A. Gerzon, “General Metatheory of Auditory Localisation,” in *92nd AES Convention*, Vienna, Austria, Mar. 1992.

[31] J. S. Abel and P. Huang, “A Simple, Robust Measure of Reverberation Echo Density,” in *121st AES Convention*, San Francisco, USA, Oct. 2006.

[32] J. S. Abel and P. Huang, “Aspects of Reverberation Echo Density,” in *123rd AES Convention*, New York, NY, USA, Oct. 2007.

[33] *DIN 18041 - Hörsamkeit in kleinen bis mittelgroßen Räumen (Acoustical Quality in Small to Medium-Sized Rooms)*, May 2004.

[34] ITU-R, *Rec. ITU-R BS.775-1, Multichannel Stereophonic Sound System with and without Accompanying Picture*, 2006.

[35] T. Lokki and J. Pätynen and V. Pulkki, “Recording of Anechoic Symphony Music,” in *Proc. Acoustics ’08 Paris*, Paris, France, Jun. 2008.

[36] *Steinberg*, URL: `http://www.steinberg.net`.

[37] M. Frigo and S. G. Johnson, “The Design and Implementation of FFTW3,” *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[38] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1975.

[39] A. Farina, “Convolution of Anechoic Music with Binaural Impulse Responses,” in *Proc. of PARMA-CM Users Meeting*, Parma, Italy, 1993.

[40] G. P. M. Egelmeers and P. C. W. Sommen, “A New Method for Efficient Convolution in Frequency Domain by Nonuniform Partitioning for Adaptive Filtering,” *IEEE Transactions on Signal Processing*, vol. 44, no. 12, pp. 3123–2129, 1996.

[41] G. Garcia, “Optimal Filter Partition for Efficient Convolution with Short Input/Output Delay,” in *113th AES Convention*, Los Angeles, CA, USA, Oct. 2002.

[42] T. Ilmonen and T. Lokki, “Extreme Filters - Cache-Efficient Implementation of Long IIR and FIR Filters,” *IEEE Signal Processing Letters*, vol. 13, no. 7, pp. 401–404, 2006.

[43] F. Wefers, “Effects of Hardware on Optimal Filter Segmentations for the Segmented Convolution,” in *2nd Joint Conference of the Acoustical Society of America and the European Acoustics Association (Acoustics’08)*, Paris, France, 2008.

[44] W. G. Gardner, “Efficient Convolution without Input/Output Delay,” *JAES Journal of the Audio Engineering Society*, vol. 43, no. 3, pp. 127–136, Mar. 1995.

[45] A. Xu et al., “Real-Time Streaming of Multichannel Audio Data over Internet,” *J. Audio Eng. Soc.*, vol. 48, no. 7/8, pp. 627–641, 2000.

[46] L. Dagum and R. Menon, “OpenMP: An Industry Standard API for Shared-Memory Programming,” *IEEE Computational Science & Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[47] *Best Service*, URL: `http://www.bestservice.de/`.