UNIVERSITY OF VICTORIA
Department of Electrical and Computer Engineering

ELEC 407
DSP Project

Algorithmic Reverberation

Combining Moorer's reverberator with simulated room IR reflection modeling

Project Team:

Hudson Giesbrecht

Will McFarland

Tim Perry

Prepared for:

Michael McGuire

July-Aug, 2009

Hudson Giesbrecht Will McFarland Tim Perry

Abstract

Recorded sound often sounds dry and flat. A reverberator adds the impression of ambience to an acoustic signal that was recorded in a “dry” environment – the ultimate goal is to simulate a room’s natural reverb. This paper presents basic acoustic theory as well as an implementation of Moorer’s reverberator. Additionally, a hybrid algorithmic reverberator is presented, using simulated room impulse responses for early reflection modeling in place of the traditional tap-delay line.

CONTENTS

Algorithmic Reverberation – A Hybrid Appoach

Combining Moorer's reverberator with simulated room IR reflection modeling

1. OVERVIEW OF REVERBERATION

Clap your hands in a large room; the initial impulse of broadband noise conceptually resembles a delta function. The resulting reflections and ambience that you hear after exciting the room, is reverb. Reverberation is the indirect sound that reaches a listener from a source. From any source there is sound that reaches the listener in a direct path, as well as sound that reaches the listener indirectly through reflections in the acoustic space. This reflected sound reaches the listener later then the direct sound due to its longer travel path. It has been diffused by certain surfaces, and lost energy due to propagation through the air and absorption in the room. The reflected/diffused sound continues to interact with its surroundings, until it has been fully absorbed.

1.1. Early Reflections and Late Reverb

Reverb can be separated into two main components, which can be viewed when looking at the impulse response representation of a location in room:

1. **Early Reflections** – the first reflections that we hear within about 100ms of hearing the direct sound of the source.

2. **Late Reverberation** – the reverberant sound field after about 100ms, until it fully decays. Late reverb is characterized by a dense texture of diffused reflections that reach our ears from many different paths. These diffused reflections are out of phase with one another, causing us to hear the comb filtering effect. We perceive this as ambience.

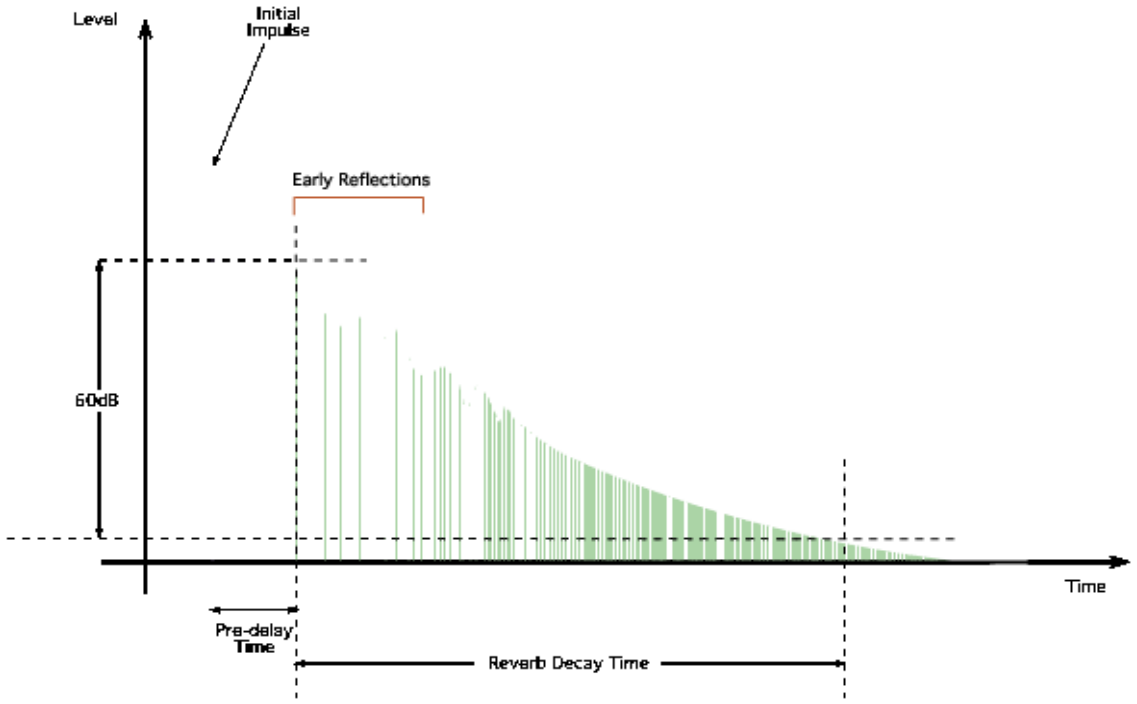

Figure 1: Typical impulse response of a room, highlighting the ITDG, early reflections, and RT60 [F1].

Two acoustic parameters will be frequently discussed in this paper; viewing a typical impulse response in the time domain, the relationship between these parameters and the two components of reverb (early reflections and late reverberation) becomes apparent.

- **Initial Time Delay Gap (ITDG)** – the time gap between the arrival of the direct sound that we hear, and the first early reflection. This gives us an impression of intimacy in relation to walls in a room, and helps our ears reveal our relative position within that room.

- **RT60 -** the *reverberation time, RT60,* is the time it takes for the acoustic signal to decay by 60dB. The reverb time varies throughout the spectrum, and can be measured at discrete frequency values. For practical applications, averaging schemes are used to define the reverb time in the portion of the spectrum where human hearing is most sensitive (between about 125 Hz and 4000 Hz is where we perceive the most significant sound colouration by reverb). RT60 is defined according to the *Sabine* and *Eyring* equations [1] as a function the volume of a room and the absorption of sound energy inside that room.

1.2. Artificial Reverberation

Many musical recordings are done in a way that mostly the direct sound is recorded, with little natural reverb. Playback of such a recording sounds dry and lacks the spaciousness that listeners are accustomed to hearing. The purpose of artificial reverb is to add the impression of ambience to an acoustic signal that was recorded in a "dry" environment – the ultimate goal is to simulate natural reverb.

A wide range of filtering algorithms have been developed that do recreate elements of reverb; this report will document the implementation of a traditional reverb algorithm, and a modified version that takes advantage FFT convolution. Few reverb algorithms are successful at accurately simulating a real acoustic space. The most practical method of simulating an acoustic space with a very high level of realism is to convolve a dry signal with the impulse response of a real acoustic environment, typically a room. This is known as convolution reverb.

The reverb model that is designed here is purely algorithmic. As a result, the goal is not necessarily to simulate an existing acoustic space, but to create a virtual acoustic space. However, the modified reverb design does model early reflections using convolution with a simulated room impulse response (RIR). This was generated using an algorithm, and the overall result can be viewed as a hybrid between traditional reverberation algorithms and convolution reverb.

2. INITIAL IMPLEMENTATION: MOORER'S-TYPE REVERBERATOR

2.1. Expanding on Schroeder's Reverberator

By employing algorithmic reverb models of higher complexity, a more convincing reverb (more natural sounding) can typically be achieved. Original implementations of reverb algorithms focused on late reverberation. By adding a tap delay line to simulate early reflections, Moorer expanded on Schroeder's reverb model by simulating two stages of reverb. Moorer's reverberator consists of a FIR stage with 10-20 taps to create the early reflections, followed by a stage of parallel comb filters and cascade all pass filters to simulate the dense late reflections of the reverb tail.

Stage A

- Tap delay line FIR network to simulate early reflections

Stage B (Schroeder's Reverberator)

- Parallel comb filters followed by first order lowpass filters to simulate a smooth decay with high frequency roll off as time progresses
- Allpass filter to increase echo density without adding colouring to the magnitude frequency response.

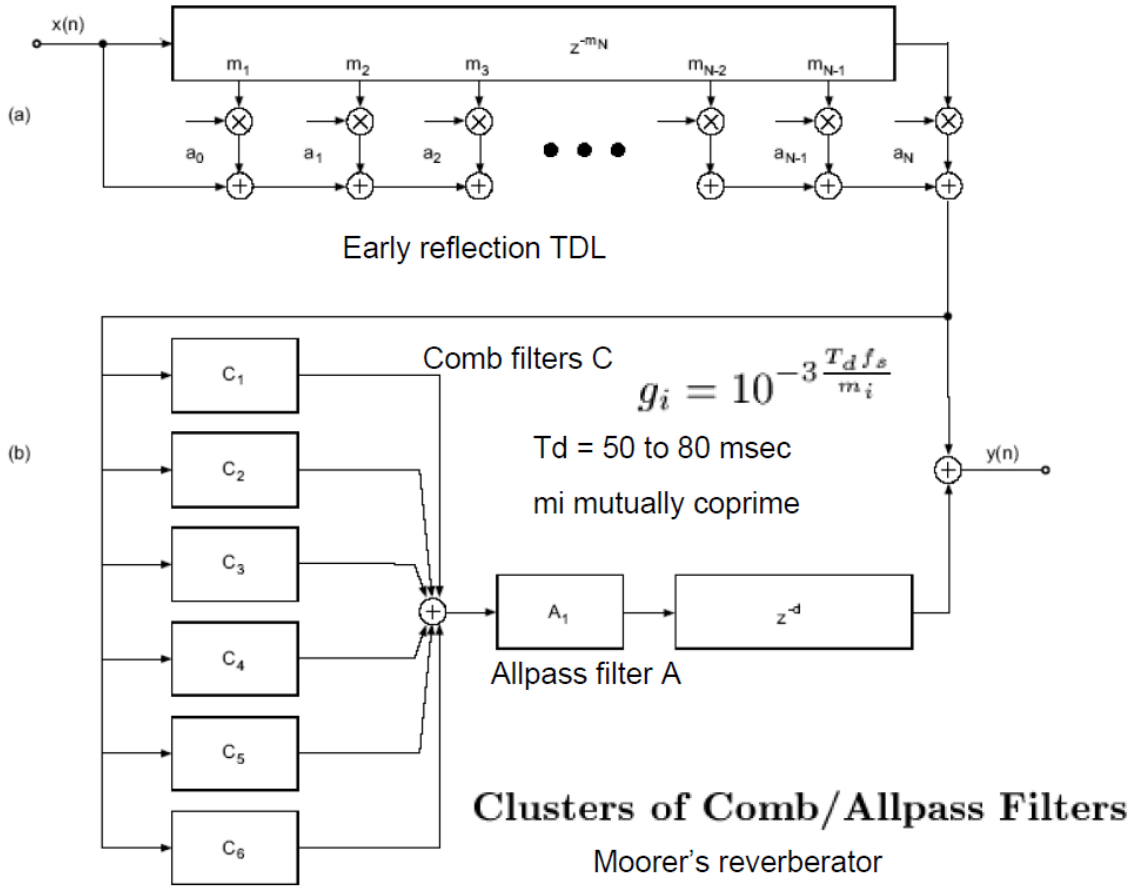

Figure 2: Moorer's reverberator block diagram [3]

1.1 Stage A: Tap Delay Line

A tapped delay line sums copies of the input signal that have been increasingly delayed. The output of the delay line simulates the sparse early reflections that happen in the first 100msec of reverberation.

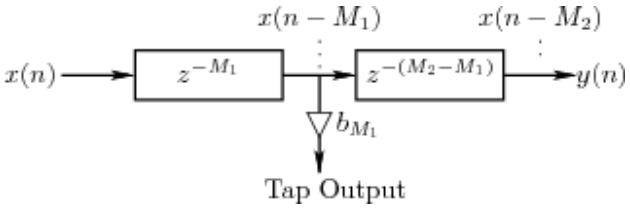

Figure 3: Tap delay line block diagram [1]

2.1 Stage B: Parallel Comb Filters Feeding Allpass Filters

The late reverb simulation in Moorer's Reverberator is based on the earlier work of Schroeder. The early reflections from the tapped delay line are first fed into a stage of parallel comb filters , which are followed by one or a series all pass filters. This is implemented with IIR filters, resulting in a computationally efficient way to simulate the reverb tail. Additional colouration was added at this stage: lowpass filtering was used to simulate high frequency roll off as time progresses and the reverb tail settles.

Parallel Comb Filter Bank

The parallel feedback comb filters increase the density of the individual delays generated by the tapped delay line simulating an increase of individual reflections. Comb filters get their name from the appearance of their combed amplitude response which also serves to color the reverb.

Feedback (IIR) comb filters are implemented by the difference equation:

$$y(n) = cx(n) - gy(n - M) \quad (1)$$

The delay values g for the comb filters should be mutually prime to one another; flutter echoes are quite audible if this is not followed.

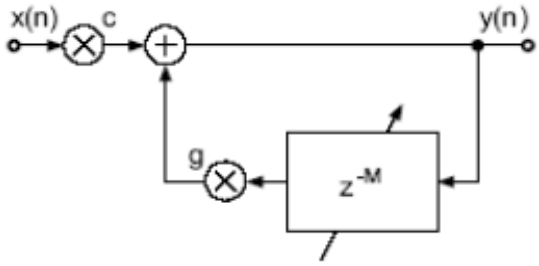

Figure 4: IIR comb filter block diagram. M represents the number of samples of delay, as seen in the difference equation.

For the usual case of c=1 the transfer function of the comb filter is calculated to be:

$$H(z) = \frac{1}{1 - gz^{-M}}$$

The corresponding amplitude response is:

$$G(\omega) = \left|H\left(e^{j\omega}\right)\right| = \frac{1}{\left|1 - ge^{-j\omega M}\right|}$$

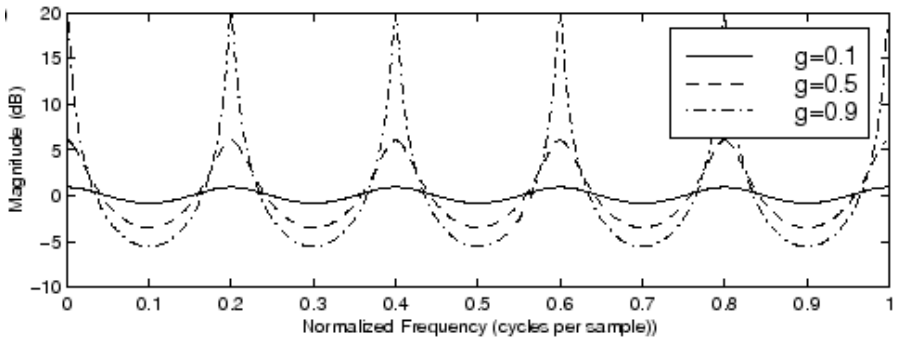

Figure 5: Amplitude response of a feedback comb filter [3]. M=5, c=1, and g =[0.1 0.5 0.9].

The peaks and valleys in the amplitude response impart a metallic sound to the output of the comb filters. Low pass filters inserted into the feedback loop simulate air absorption of high frequency sound and remove the metallic coloration of the comb filters.

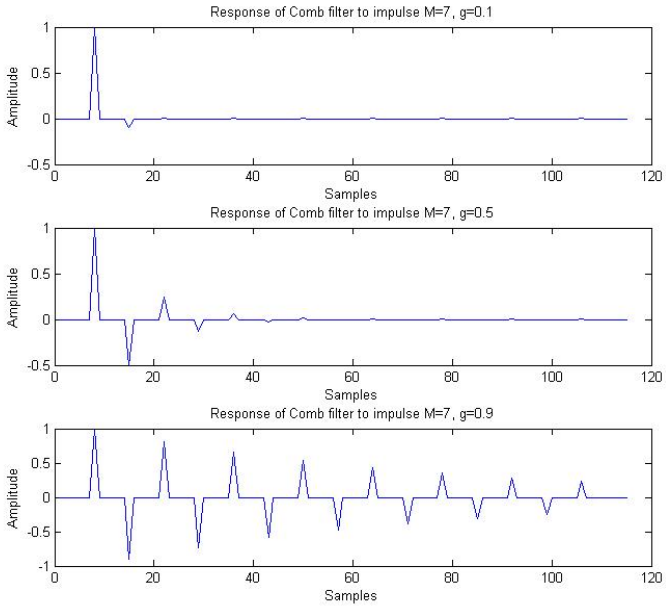

Figure 6: Comb filter impulse response for M=7, g=0.1, 0.5, and 0.9 respectively

Allpass Filters

Allpass filters rapidly increase the density of the reflections to simulate the late reverb. Like feedback comb filters, allpass filters expand a non zero input into an infinite impulse response.

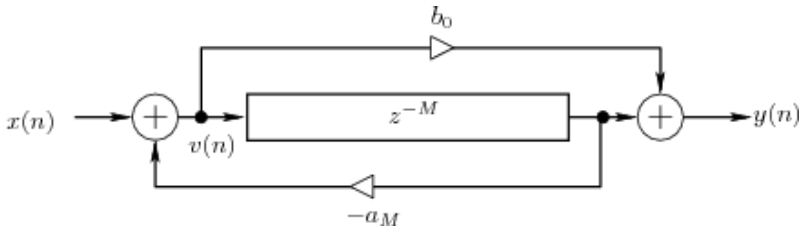

Figure 7: Combination IIR/FIR comb filter becomes an allpass filter when $b_0=a_M$

Allpass filters are implemented by the direct form 1 difference equation:

$$y(n) = b_0 x(n) + x(x - M) - a_M y(n - M)$$

This gives a transfer function of:

$$H(z) = \frac{b_0 + z^{-M}}{1 + a_M z^{-M}}$$

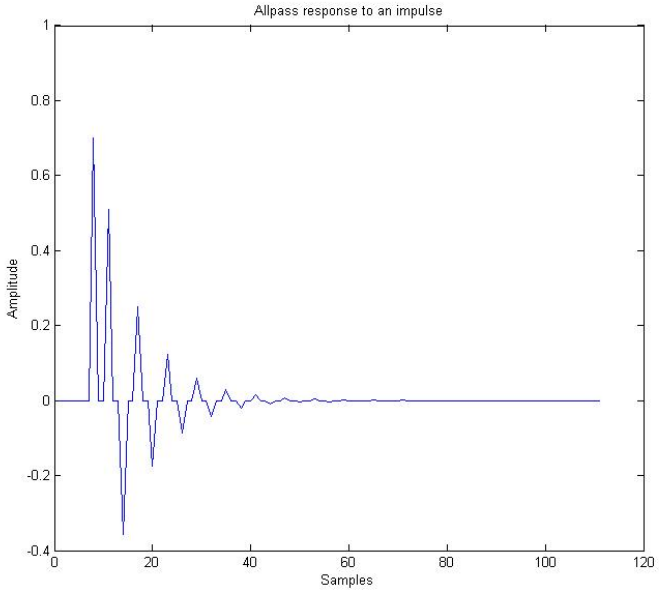

Figure 8: Allpass filter impulse response. M=3, g=0.7

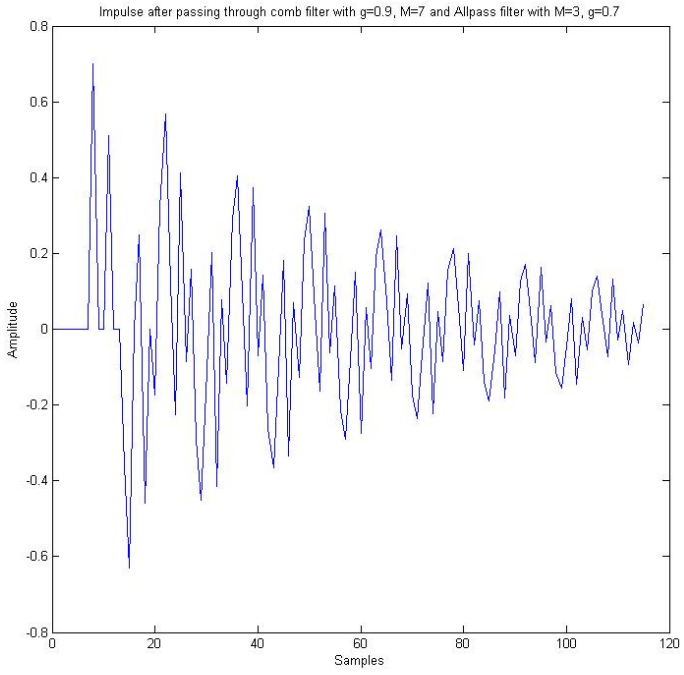

Figure 9: Response to unit impulse input after comb filtering, followed by allpass filtering. Comb filter parameters: M=7, g=0.9. Allpass filter parameters: M=3, g=0.7

3. REVISED DESIGN: CONVOLUTION WITH SIMULATED ROOM IR

3.1. Modeling Early Reflections with a Simulated Room Impulse Response

The most natural sounding reverberation is convolution reverb with a measured room impulse response (MRIR). But what if we can simulate a basic room to take care of the critical early reflections, and in doing so enhance the realism by adding a familiar spatial dimension to algorithmic reverb reverb? A potentially more realistic sounding algorithm could make use of a low order FIR filter that models a simple room; early reflections could be created using convolution with this simulated, RIR rather than a tap delay line. Intuitive control of reverberation parameters would be based on real acoustic attributes such as: room size, listener location in room, source location in room, and broadband absorption of sound energy in the room.

By taking advantage of the existing algorithm for late reverberation, a hybrid reverb can be created that is less computationally intensive than convolution with a measured room IR (depending on the length of the measured IR vs the simulated IR). The hybrid algorithmic reverb that was implemented can be conceptually explained as follows:

- Replace the Tap Delay Network from Moorer's reverberator with an FIR filter that simulates the impulse response of a basic rectangular room model.
- Convolve the dry signal with the simulated room IR to obtain a more realistic representation of the early reflections. For added speed, FFT convolution was used.
- Feed a mix of the convolved signal with early reflections and dry signal into the late reverb stage (Stage B).

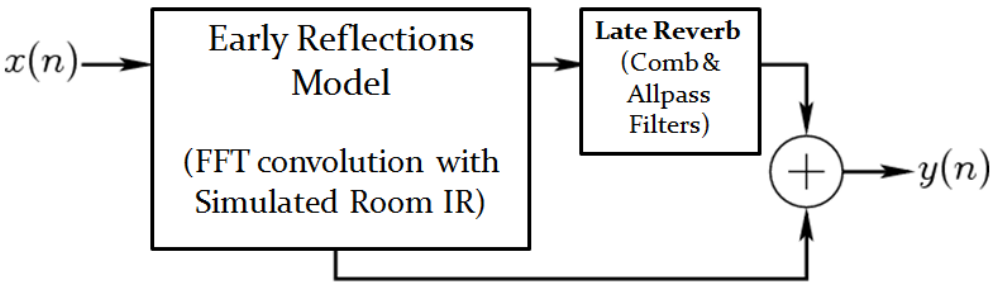

Figure 10: Hybrid reverb block diagram, illustrating two two main stages.

In order to realize this idea, a room model was first needed. A basic rectangular room modeling algorithm was developed in [6]. This algorithm simulates a simplified room as a low order FIR filter. The order of the filter is governed by the number of successive reflections that are modeled for a given echo in the room. Basic flat walled room dimensions were used, and diffusion is not well represented by this model, resulting an *echo density* that sounds un-natural if larger room sizes are used. For small rooms, however, this model does an excellent job of providing the initial time delay gap that we naturally cue into when listening to sounds in physical acoustic environments.

The basic model is summarized below, based on [6]:

The distance to each source in the room can be expressed

$$d_{ijk} = \sqrt{{x_i}^2 + {y_j}^2 + {z_k}^2} \tag{2}$$

The unit impulse function of each virtual source is delayed by a constant, which is determined by the source distance and the speed of sound c. The unit impulse function for a source then takes the form:

$$u_{ijk}(t) = t - \frac{d_{ijk}}{c} \tag{3}$$

Defining additional parameters to represent the relative weighting of reflections:

$$e_{ijk} = b_{ijk} r_{ijk} \tag{4}$$

is the magnitude of each echo, and *r* is the total reflection coefficient of the surface (inverse of the absorption coefficient). Finally, for the creation of the transfer function, we need to know if a particular reflection should be present. The following parameter is defined:

$$a_{ijk}(u_{ijk}) = \begin{cases} 1, & \text{if } u_{ijk} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The transfer function is simply the sum of these echos (delayed, weighted impulses) in three dimensions:

$$h(t) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} \sum_{k=-n}^{n} a_{ijk} e_{ijk} \tag{6}$$

A more detailed overview of this transfer function representation can be found in [6]. The function used for implementing the IR is also included in the appendix

3.2. Parameterization for Simulated Room IR

A rectangular room was simulated to have similar acoustic parameters to an ideal recording studio control room, according to the specifications in [3]. To simulate the soundfield in the rectangular room, absorption coefficients for the wall materials first needed to be assigned. RT60 is a function of the room volume and the average absorption of each surface, as described by the Sabine (left) and Eyring (right) equations [9]:

$$\frac{0.161V}{S_{tot}\alpha_{sab} + 4\ \text{mV}} = T_{60} = \frac{0.161V}{S_{tot}[-2.3\ \log(1 - \alpha_{ey})] + 4\ \text{mV}} \tag{7}$$

where V is the volume of the room, S_{tot} is the total surface area, and α_{sab} and α_{ey} are coefficients that represent the average absorption of all surfaces.

The parameters for the simulated IR were chosen by applying suitable absorption coefficients to RoomSim [5]. The script takes into consideration many test and environmental parameters; some parameters are well defined, several have been estimated. The volume of the modeled room is 263 m^3, intended to give a relatively short reverb time across the spectrum. The intention is to start with a basic, controlled reverb that retains some of the original signal clarity. While the objective is not to design precisely to specifications, a guideline is helpful so that we may aim for compatibility with the existing reverb model.

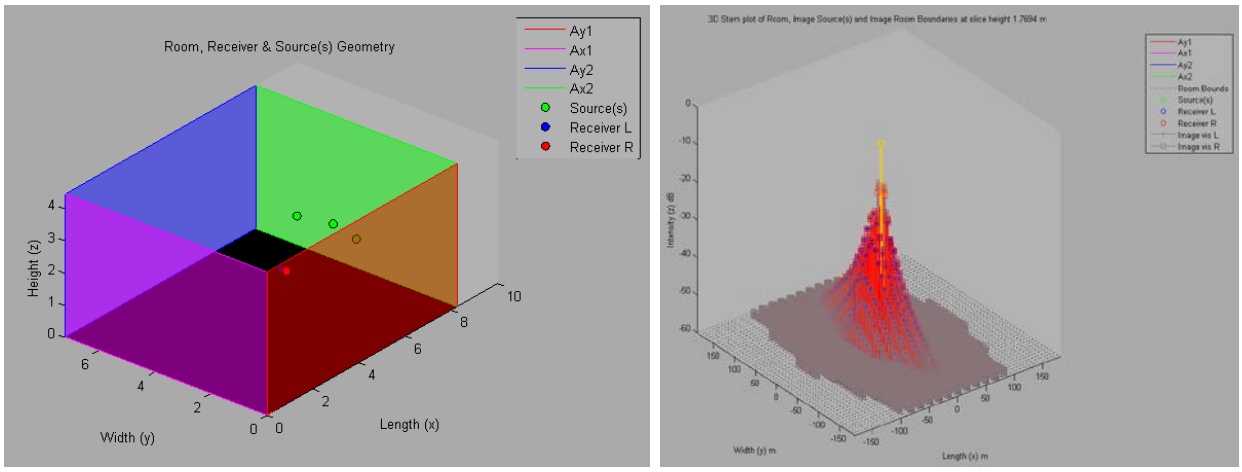

Figure 11: Room Sim [5] simulation layout (left) with 3 monitor speakers (30 degrees of separation between monitors) at a 2.2m radial distance, angled 10 degrees down toward the listening position for sources. Two simulated omnidirectional test microphones separated by 14.5 cm are located in the listening position as sensors (listening position used is at 38% of the length dimension back from the front wall, centered width wise at a height of 1.2m). Perceived location of early reflections (right) plotted for room layout in Room Sim [5], stem height is sound intensity height.

In attempt to model the acoustics (primarily, the early reflections) of the desired room, the absorption coefficients in Table 1 (plotted in Figure 12) were used in the simulation. The goal was to use sidewalls with no acoustic treatment, but some acoustic treatment on the front, back, and top of the room. The surfaces that are intended to be partially treated are the back wall (the "live end", a diffusive surface) and the front wall (referred to as the "dead end" a control room, a surface with high absorption in most places). To simplify the parameterization when implementing the room model in a hybrid reverb, a single reflectivity coefficient was chosen.

If r is the reflectivity coefficient of all surfaces in the room, we can relate it to α, the average selected absorption coefficient of the room, as follows:

$$\alpha = 1 - r^2 \tag{8}$$

Table 1: Absorption coefficients used in RT60 and IR simulation

Surface	**Main Surface Material Simulated**	**125 Hz**	**250 Hz**	**500 Hz**	**1000 Hz**	**2000 Hz**	**4000 Hz**
Back Wall (Ax1)	RPG Skyline diffusor (attenuation at 125 Hz added)	0.15	0.34	0.28	0.29	0.19	0.16
Front Wall (Ax2)	hypothetical 50% broadband attenuation (acoustic foam and glass)	0.75	0.75	0.75	0.75	0.75	0.75
Side Wall 1	gypsum wallboard	0.3	0.1	0.05	0.04	0.07	0.1

(Ay1)							
Side Wall 2 (Ay2)	gypsum wallboard	0.3	0.1	0.05	0.04	0.07	0.1
Floor (Az1)	varnished cork parquet on joists (floating)	0.15	0.11	0.10	0.07	.0.06	0.7
Ceiling (Az2)	acoustic tile (suspended)	0.5	0.7	0.6	0.7	0.7	0.5

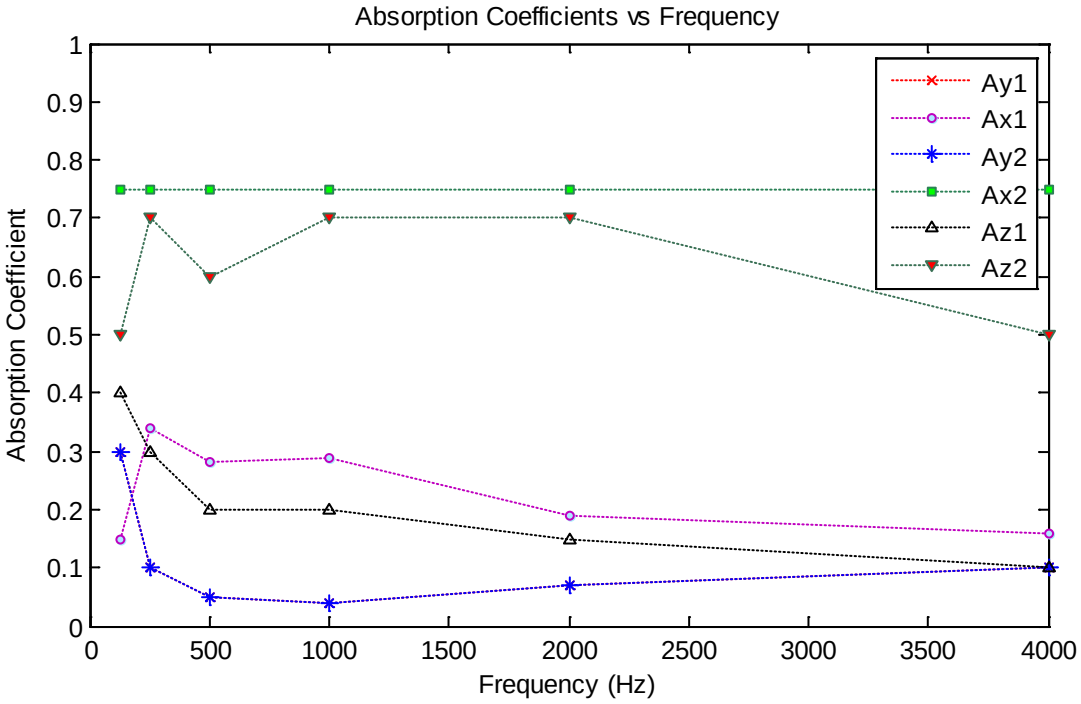

Figure 12: Absorption coefficients versus frequency for simulated surface materials

The RT60 simulation was run using a generalized version of the Eyring equation at six frequency bands, taking into account environmental parameters such as room temperature, humidity and air density (typical values were used). The results are displayed in Figure 13. These finer details were not included in the implementation, but could be useful if an advanced room simulation reverb algorithm was being developed (particularly if used for more than simply early reflection modeling). The purpose of this simulation was to approximate a reverberation time to use as a design parameter.

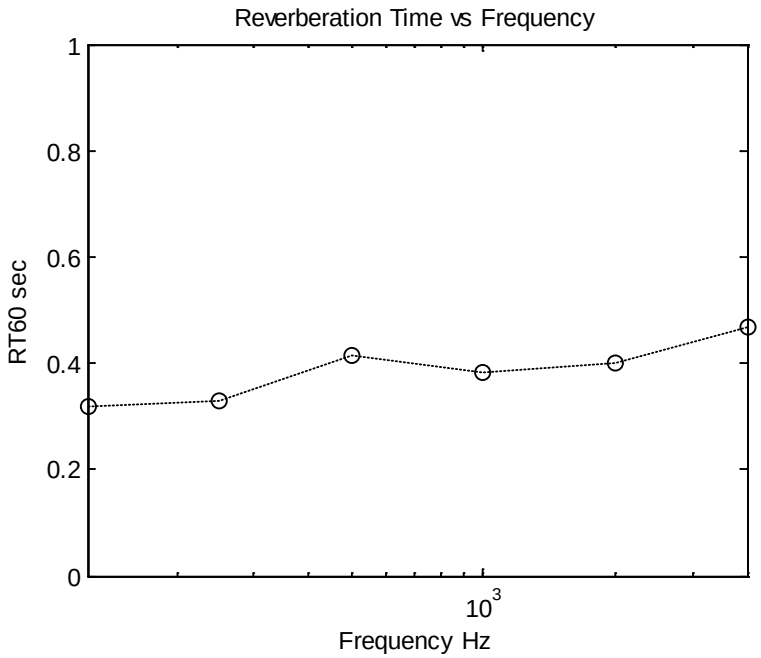

Figure 13: RT60 versus frequency for simulated rectangular room with estimated absorption coefficients.

The values of the estimated RT60 coefficients are between 0.3 and 0.5 seconds throughout the hearing spectrum. Having an estimated RT60 range is important, as we need to combine this modeled room IR with the late reverb model. For a constant spatial impression they should both exhibit a similar reverb time. This is one design factor that can help the blending between early reflections and late reverberation seems less artificial. In practice, however, achieving an impression of continuity between the reverb stages was challenging.

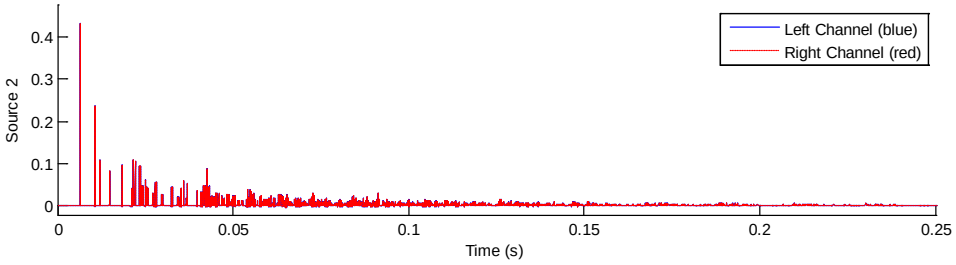

Figure 14: Impulse response for rectangular room model with selected coefficients, plotted for center sound source.

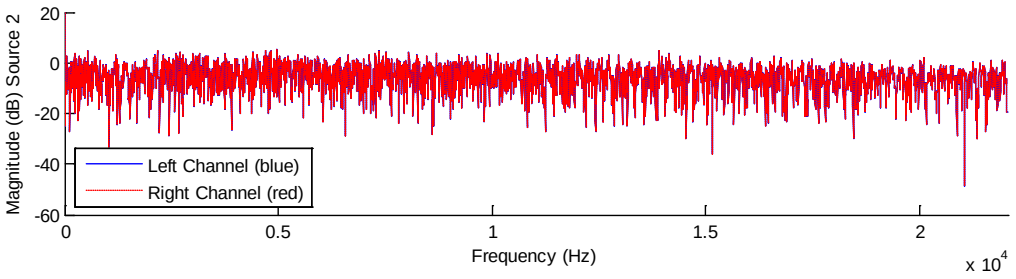

Figure 15: Frequency response in the rectangular room model with selected coefficients, plotted for center sound source.

3.3. Implementation using High Speed Convolution

Without an efficient method of performing the convolution with a simulated room IR, the new early reflections stage for the hybrid reverb is little more than a novelty. Audio signals are typically of sufficient length to render direct time domain convolution impractical. Computation time becomes ridiculous with longer signals, and real-time implementation is not possible. An alternative is *High Speed Convolution* using windowed FFTs and frequency domain multiplication. FFT algorithms have a computational complexity of O(n log n) versus O(n^2) for the direct application of a DFT. The maximum efficiency is achieved if n is a power of 2.

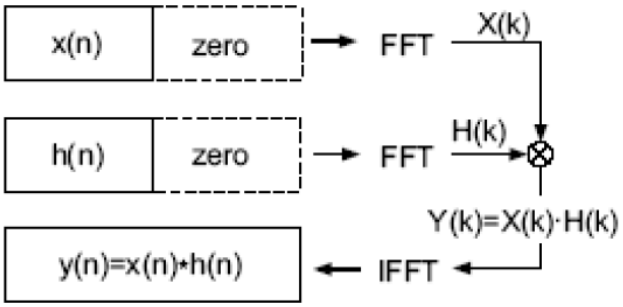

Figure 16: High Speed/Fast Convolution [3]

The result that we want to obtain can be computed by calculating the product of RIR's FFT and the input signal's FFT, and taking the IFFT of the result:

$$y(n) = IFFT\{FFT\{h(n)\} \bullet FFT\{x(n)\}\} \quad (9)$$

However, we can typically do this faster by first chopping up the longer signal. FFT convolution with rectangular windowing and overlap-add was implemented for the convolution of dry source $x(n)$ with a simulated room IR, *h(n)*. The input signal was segmented into blocks, and *h(n)* was defined as the filter. Convolution was performed in the frequency domain on successive blocks of the input signal with *h(n)*. Taking *Nx* to be the length of *x(n*) and *Nh = N +1* to be the length of *h(n)*. The procedure was as follows [3]:

- The filter *h(n)* was zero padded to length *NFFT = 2N = 2(Nh-1)*. Next, the zero padded filter's FFT was taken, yielding *H(k)*.
- The dry signal *x(n)* was segmented into blocks $x_i(n)$, of length *N*. Each of these blocks was zero padded to a length *2N = NFFT*.
- The FFT of each zero padded frame was taken, yielding $X_i(k) \quad with\ k = 0, 1, 2, \dots 2N - 1.$
- Frequency domain multiplication was applied:

$$Y_i(k) = X_i(k)H(k)$$

- The IFFT was taken of each $Y_i(k)$.
- The convolution results were overlap added, yielding the output signal *y(n)*. The length of the output signal is *Nx + Nh – 1*.

An example Matlab script for performing the above convolution can be found in the Appendix.

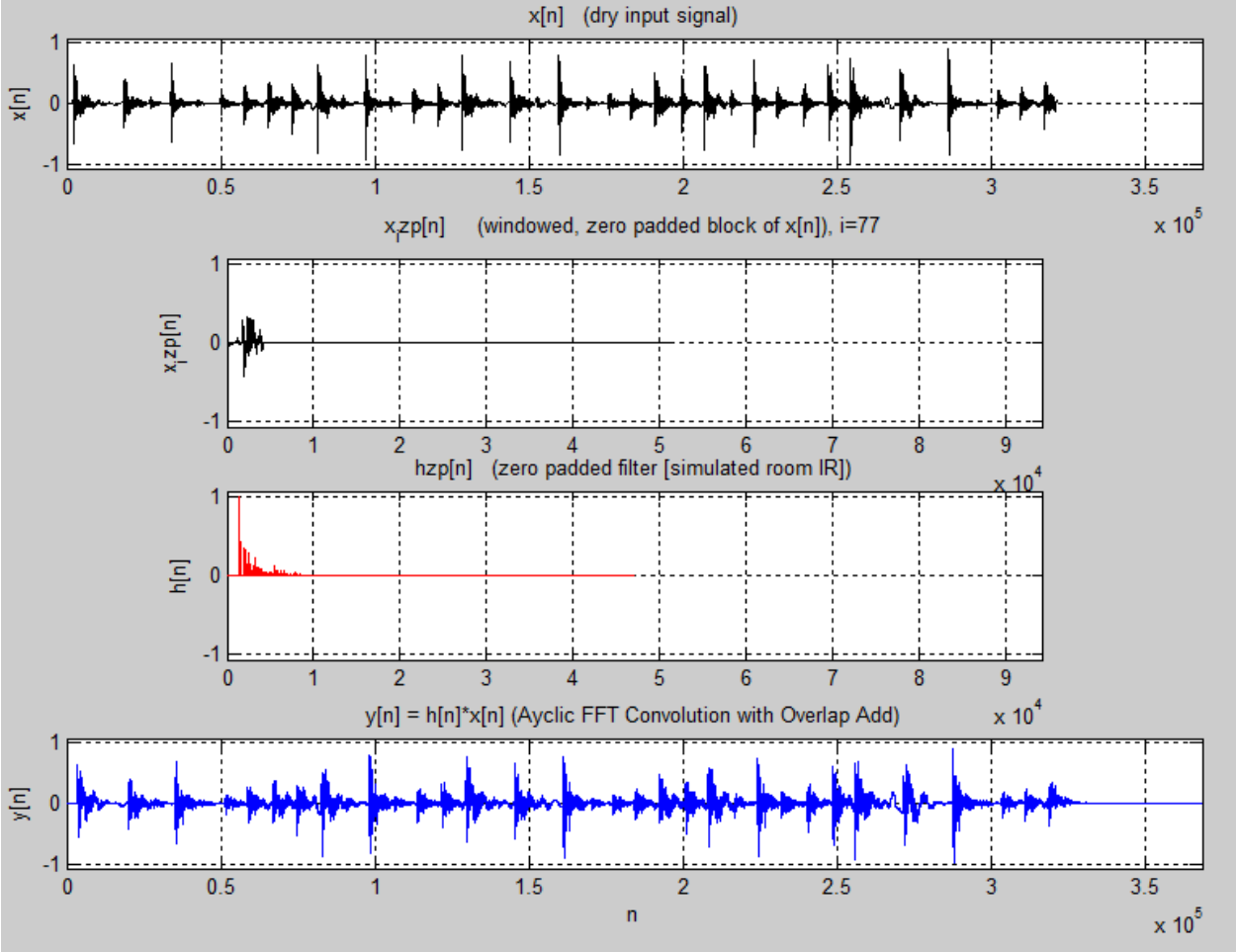

Figure 17: Fast convolution of a dry input signal with modeled room IR. Dry source (top), last windowed block of dry source (2nd down), zero padded IR (3rd down), output (bottom). The input source audio is a Cajun percussion clip.

By implementing acyclic convolution using cyclic convolution with zero-padding, the FFT can be used to perform a cyclic convolution when its length N is a power of 2 [8]. Using zero padding in the time domain results in more samples and tighter spacing in the frequency domain (like a higher sampling rate will achieve). The resulting number of nonzero output samples will be at most $Nx + Nh - 1$.

The motivation to embed and acyclic convolution into a zero-padded cyclic convolution (*High Speed or Fast Convolution*) is that a cyclic convolution is very efficient to compute. However, if insufficient zeros are added with this method, the same problem as encountered with cyclic convolution will occur: some convolution terms will wrap around and add back onto earlier terms, causing time domain aliasing.

This method of FFT convolution yields the same result as a linear convolution, but is much more computationally efficient for long signals (and hence faster). For filter kernel lengths over about 50 to 80 samples, depending on the hardware, FFT convolution is faster than standard convolution. In practice, precision of a convolution result (assuming a correct convolution without aliasing) is determined by the

speed of the calculation. This is due to round-off error in the computation, which is proportional to computation time [8].

An alternative method of FFT convolution was initially used, which employed raised-cosine windows of length *WLen = NFFT = 4096* samples and a hop size of *R = WLen/4* for windowing the input signal and overlap-adding the convolved segments. This method was significantly less efficient; it involved segmenting the IR in addition to the input signal, since the IR length was longer than 4096 samples. While the FFT size was smaller, many more FFTs were required than with the above high speed convolution scheme. Additionally, it resulted in a lower frequency resolution. While the slower FFT convolution scheme was replaced with fast convolution for early reflection modeling, it continued to see use for time-frequency analysis of the reverberation results.

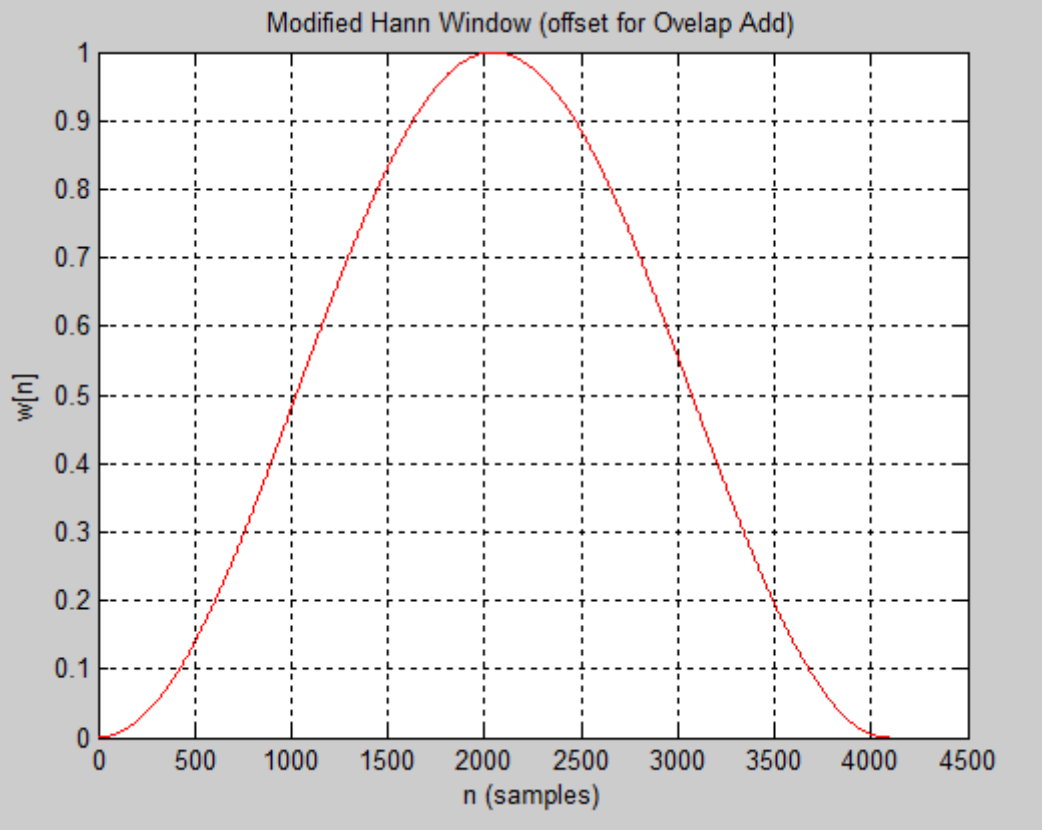

Figure 18: Raised cosine window used in original FFT convolution algorithm of early reflections modeling. This window is still used for time-frequency analysis of the results, where it is applied in a phase vocoder (separate from the reverb, and not documented here).

4. TESTING AND REVISION

The figures in the following sections were generated during various I/O tests. Both reverberators that were implemented required a lot of experimentation with parameters in order to achieve reasonable results.

4.1. Moorer's Reverberator Parameterization

Tapped delay lines simulate early reflections in a simple manner; however, determining the amplitude of these early reflections and their times is not a straight-forward calculation. Our hybrid model will take care of the early reflections. For testing while developing, however, we initially simply created a series of randomly spaced early reflections with roughly linear decay to approximately simulate these reflections, as demonstrated in the following figure.

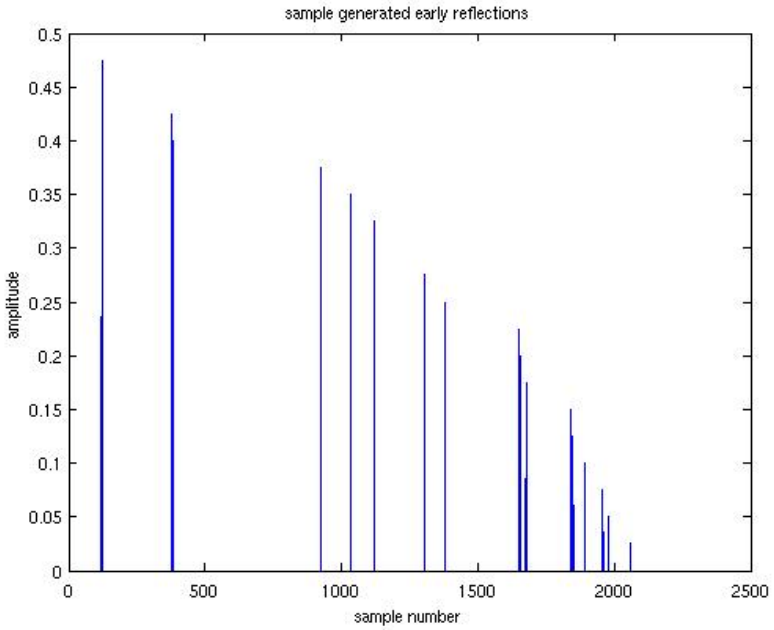

Figure 19: Randomly spaced early reflections

The quality of the reverberation, determined by the density of the reverberant sound (*echo density* or *texture*), can be tuned by varying the number of comb filters in parallel and allpass filters in series and their respective parameters. MATLAB allows for flexible alteration of component amounts, to very high levels previously thought impossible, as stated in [Schro62]: 40 comb filters with delays of 40 ms are required to generate an echo density of 1,000 echoes per second. This amount of filters is presented in the literature as clearly impossible to implement at the time of the publishing, but in MATLAB testing this configuration is a simple parameter alteration. Existing literature provides some initial values for empirical testing. [Schroe62] recommends 4 comb filters in parallel followed

by 2 all-pass filters. [Moo79] recommends 6 comb filters followed by 1 all-pass filter. The table below summarizes our findings with different configurations of all-pass and comb filters. In each test, the RT60 parameter is set to 5 seconds, so that the late reverberance is clearly audible, and the mix is set to 100% (so that only the reverberance is audible).

	1 all-pass	2 all-pass	3+ all-pass
4 comb [Schroe62]	Uneven reverb tail, appears to be 'fluttering' or pulsating.	'fluttering' still audible and signal slightly metallic sounding	Signal increasingly metallic, creates unnatural resonances
6 comb [Moo79]	Slightly less fluttering, but still audible	Late reverb flutter barely audible but slightly metallic	Metallic as above
40 comb	Very even tail without any coloration or flutter	Slightly metallic, sounds as if in a tube	Less metallic than above but sounds as if in a tube

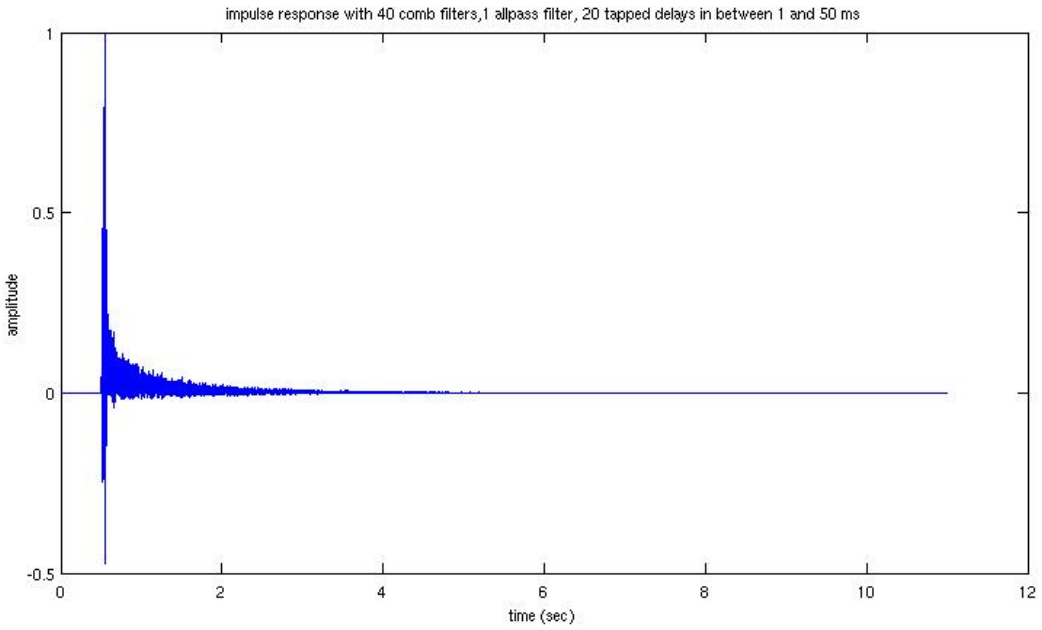

Figure 20: Impulse response of preliminary reverb implementation.

Design Improvements

While developing the reverb, we noticed that even at large numbers of comb filters the coloration from the all-pass filter(s) was very present and unnatural resonances would dominate the reverberance. Upon closer examination we noticed that our all-pass filter coefficients were incorrect:

```
y(n) = -g*x(n) + x(n-m) + g*y(n-m) (incorrect)
y(n) = g*x(n) + x(n-m) + -g*y(n-m) (correct)
```

Making this change greatly improved the quality of the reverb.

Another problem we encountered was finding the delay times of an arbitrary number of comb filters. As mentioned in [DAFX00], these delay times should be mutually coprime. We wrote a matlab excerpt to calculate N mutually coprime numbers, attached as 'coprime.m'; spread evenly across a range of numbers . After implementing this, the coloration of the reverberance decreased greatly. This is clearly because the delay of each comb filter doesn't constructively add as their delays are all coprime to each other.

The metallic quality introduced by the all-pass filters in series is because their delays are all the same amount, therefore the coloration introduced by the all-pass filter, inaudible with one all-pass filter, become audible with three all-pass filters. As with the comb filters, if the delays of the all-pass filters are coprime with each other, then the dispersion would be effective.

The comb filtering in the original implementation resulted in a pitched wine, and a metallic timber. Instead of assigning arbitrary delay values, the modified approach involved finding N coprime samples within a 0 to 30 ms range. When mapped to the base delay time of 50 ms in Moorer’s reverberator, the results cover the desired range of 50 to 80 ms.

A vast improvement to the timbre was also achieved by reducing the number of allpass filters used to just one. While allpass filters are useful for increasing echo density while reducing colouration, we can see from the spectrograms below that particularly during transients, we do get additional colouration from the allpass filters. This is due to the fact that we still have a phase response (nonlinear) with allpass filters.

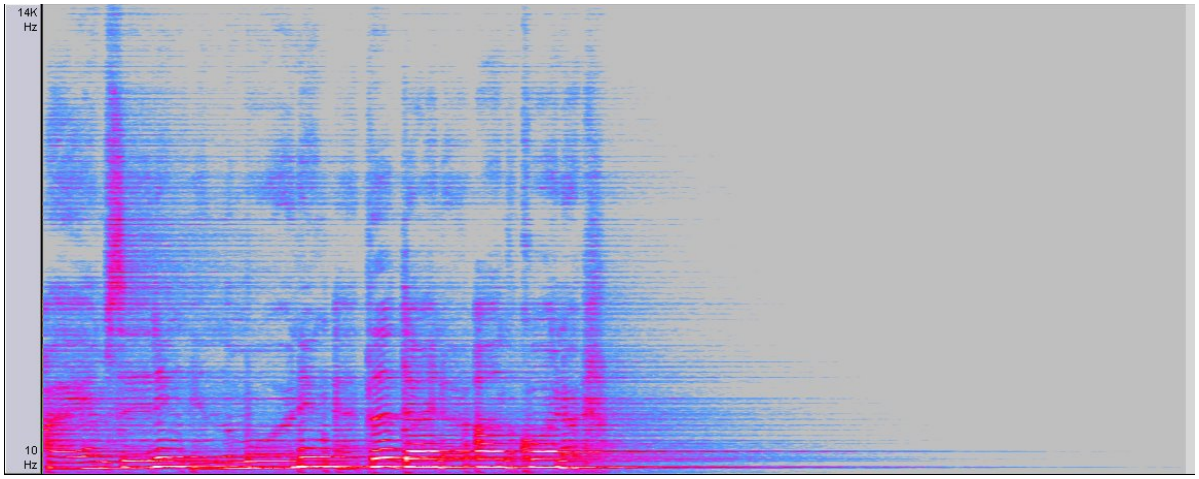

Figure 21: Spectrogram of output signal when 5 all pass filters are used in series for the Moorer's reverb implementation; input audio is diner.wav.

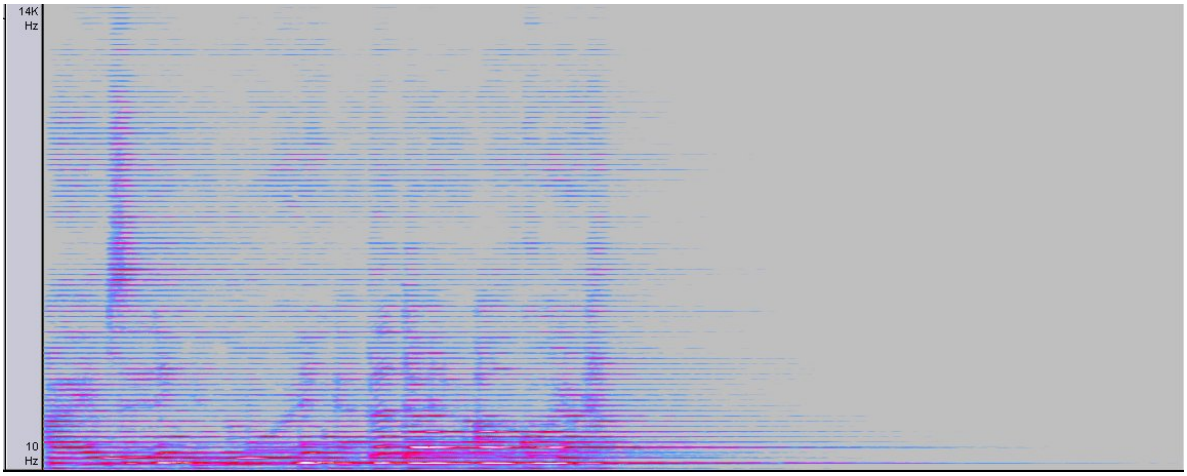

Figure 22: Spectrogram of output signal when a single allpass filter is used in the Moorer's reverb implementation; input audio is diner.wav.

Lowpass filters are normally introduced after each comb filter to simulate the roll-off of high-frequency components that naturally occurs with the reflection of sound waves. We did not implement this as after each comb filter, but instead inserted a low-pass filter after the all-pass filters in series [Staut82]. This is a feature that could be included to ensure a more natural high-frequency roll-off. As is, a first-order Butterworth filter is used because of its slow (6dB per octave) roll-off.

4.2. Hybrid Reverb

Getting the hybrid reverb to work as intended was challenging. While the two stages of reverb did work well independently of one another, integrating them together introduced some timing issues. A sample by sample timing synchronization was built into the comb/allpass filtering kernel algorithm; however, it did not work as intended. To remedy this problem, an initial time delay gap calculation for the simulated room IR was incorporated into the design. A delay compensation based on this calculation is used to line up the final late and early reverberation components based on a realistic timing relationship in acoustics. While not perfect, and not seamless, this did achieve an acceptable result. For small room simulation, however, the listening examples (discussed in the next section) do demonstrate an effective blend between the two stages of reverb. The hybrid reverb is certainly an improvement over simply convolving the input signal with the basic simulated room IR; the latter clearly models early reflections, but not ambience or high frequency roll off.

With more detailed built in parameterization, the hybrid reverb concept can be implemented to achieve a higher level of realism.

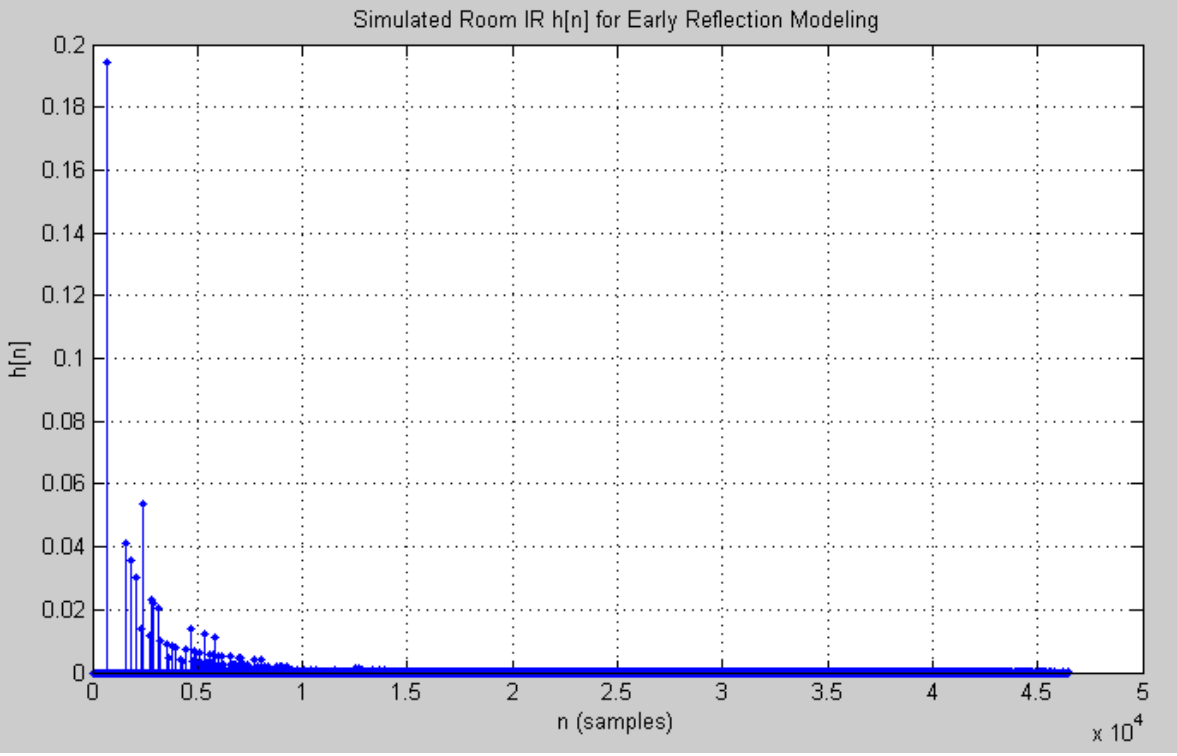

Figure 23: Modeled room IR as implemented in hybrid reverb (for single sound source). Parameters correspond to those simulated during preliminary design with *Room Sim*, with the exception of a uniform reflectivity coefficient r = 0.5. The algorithm for the rectangular room IR was developed in [6].

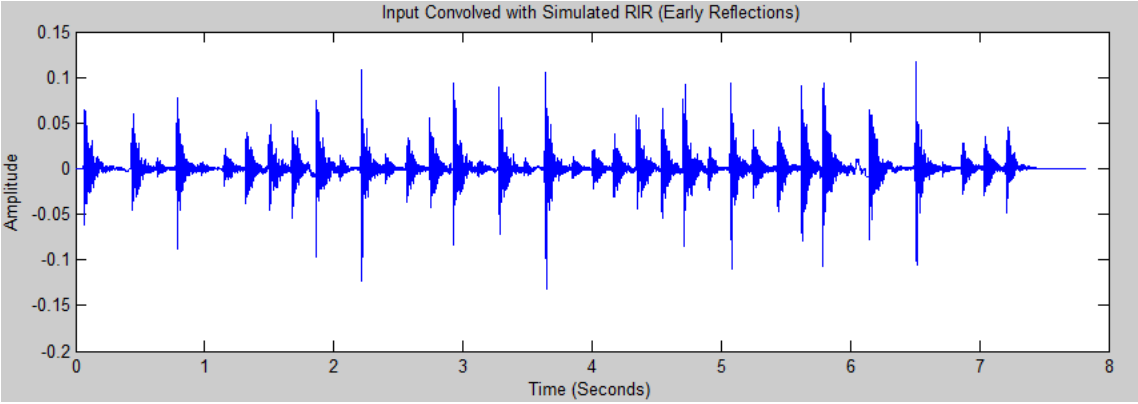

Figure 24: Output waveform of Stage A (early_reflections.m) using the hybrid reverb. A dry Cajun percussion recording has been convolved with the simulated room IR, where echoes having undergone up to 10 successive reflections have been tracked.

Example parameters used to simulate the early reflections in a small room are shown below. The room ratio used to calculate the parameters is based on Dolby's optimal room ratio for music rooms; however, extra height has been added to increase spaciousness. The modeled room was simulated using both mono and stereo microphone configurations. A spaced pair microphone configuration is implemented in the code fragment below. This configuration can produce stereo imaging through the timing differences between the two microphones, as they will be out of phase with one another for an off-axis sound source.

```
% Define room dimentions for generation of room impulse response
% Example values that use Dolby's optimum ratio for film and music rooms
% are   Mean Height = 12', Mean Width = 17.91', Mean Length = 27.76'
rw = 17.91;                 % room width
rh = 13.5;                  % room height
rd = 27.76;                 % room depth
rm = [rw rh rd ];

src = [ rw/2 rh/3 rd/6 ];   % source position in the room
src = [ rw/7 rh/3 rd/6 ];   % test imaging with source placed not central

% Stereo microphone configuration (typical spaced omni pair is modeled)
micposL = [ (rw/2 - 1.5) rh/2 rd/3 ];   % position of left mic in the room
micposR = [ (rw/2 + 1.5) rh/2 rd/3 ];   % position of right mic in the room

% n, the number of reflections to track (ex. n=6 means echoes that have
% undergone up to 6 reflections will be accounted for)
n = 10;
% r, reflectivity coefficient of the walls/floor/cieling
r = 0.5; % 0.3 is roughly a plywood sheet
```

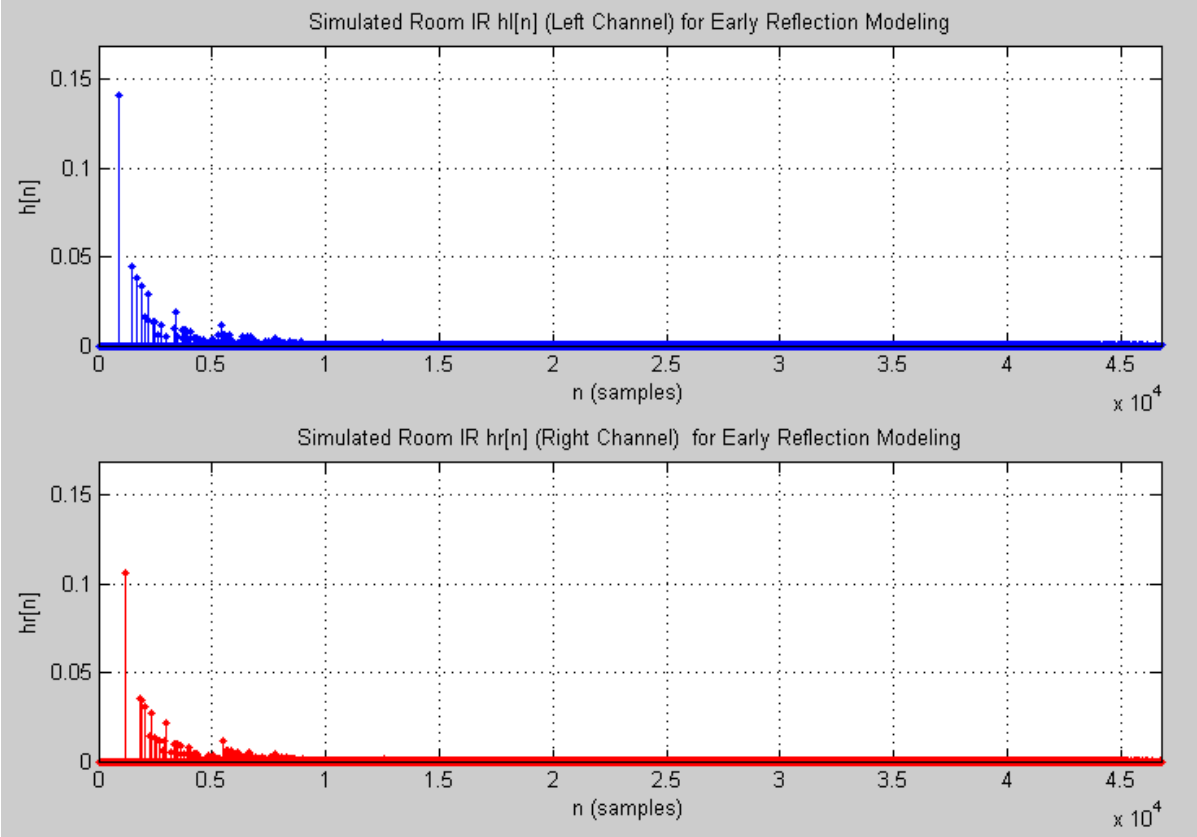

Figure 25: Implementing simulated room IRs in stereo. Two omnidirectional microphones are represented; for this spaced pair configuration, stereo imaging is the result of purely timing difference between the two simulated microphones (they are out not in phase).

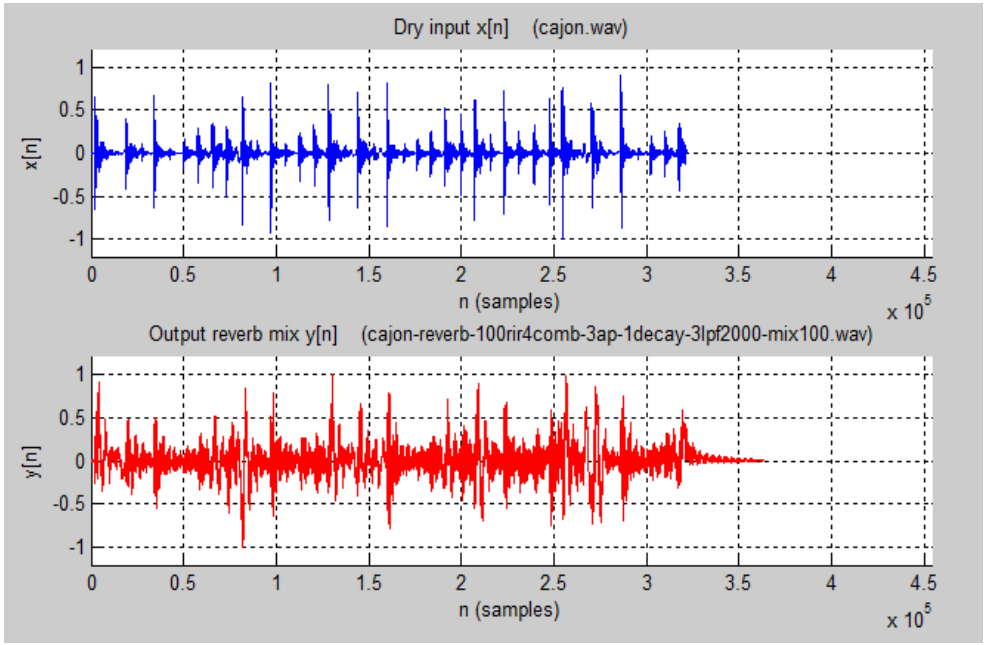

Figure 26: Example input and output waveforms for hybrid reverberation.

5. COMPARING ALGORITHMS WITH CONVOLUTION REVERB

Comparing the time frequency representation of the implemented algorithmic reverberator versus convolution reverb with a measured room impulse response, several observations can be made. While the convolution reverb models the acoustics of a real room to a high degree of accuracy, it falls short when we look outside of the human hearing range. Since frequencies below 20Hz were not measured, they are not well properly represented in the convolved signal. Figure 27, however, shows that the dry recorded signal (a Cajun percussion clip) did have significant energy in the low octave bands. The Algorithmic reverberators do capture this low end, and in certain cases it becomes over emphasized due to lowpass filtering to simulate high frequency roll-off.

If we compare the convolution reverb audio clips to the file *50_comb_40ap_1decay_10lpf1000_mix50.wav*, we notice/feel more low end thump in the algorithmic reverb case. However, the quality and realism of the reverb is not a match for convolution reverb. The lack of energy below 20Hz is often desirable, as during mastering it is common practice to high pass filter starting at about 30 to 40Hz in order to improve the clarity of the sound (other bass frequencies are emphasized to make up for the low end energy loss). We don't hear frequencies below about 20 Hz (we only feel them), so the resulting filtering of these low frequencies is hardly noticeable under most playback conditions. The frequency range where we perceive low end energy is typically between about 60 and 120 Hz, and the convolution reverb has accentuated this frequency range.

If we compare the results of the hybrid reverb *(cajon-reverb-20taps1-50_comb_40ap_1decay_10lpf1000_mix50-1)* to a relatively dry convolution reverb clip, there is actually a similar character between the early reflection dominated reverb. For small room simulation, the hybrid reverb with a very basic rectangular room model for early reflections simulation does quite a good job. While it does not perfectly simulate a room, it does communicate the character of an acoustic space.

The following waterfall (time-frequency representation) plots were created by first using a phase vocoder to analyze the amplitudes and phases of the input and output. Each FFT frame was then plotted on a time frame corresponding to sample index of each hop of the kernel algorithm. An FFT size of 4096 was used, giving fairly high frequency resolution at the cost of tracking rapid changed in the spectrum due to transients. For viewing reverb across the spectrum, a high frequency selectivity communicates the data most clearly.

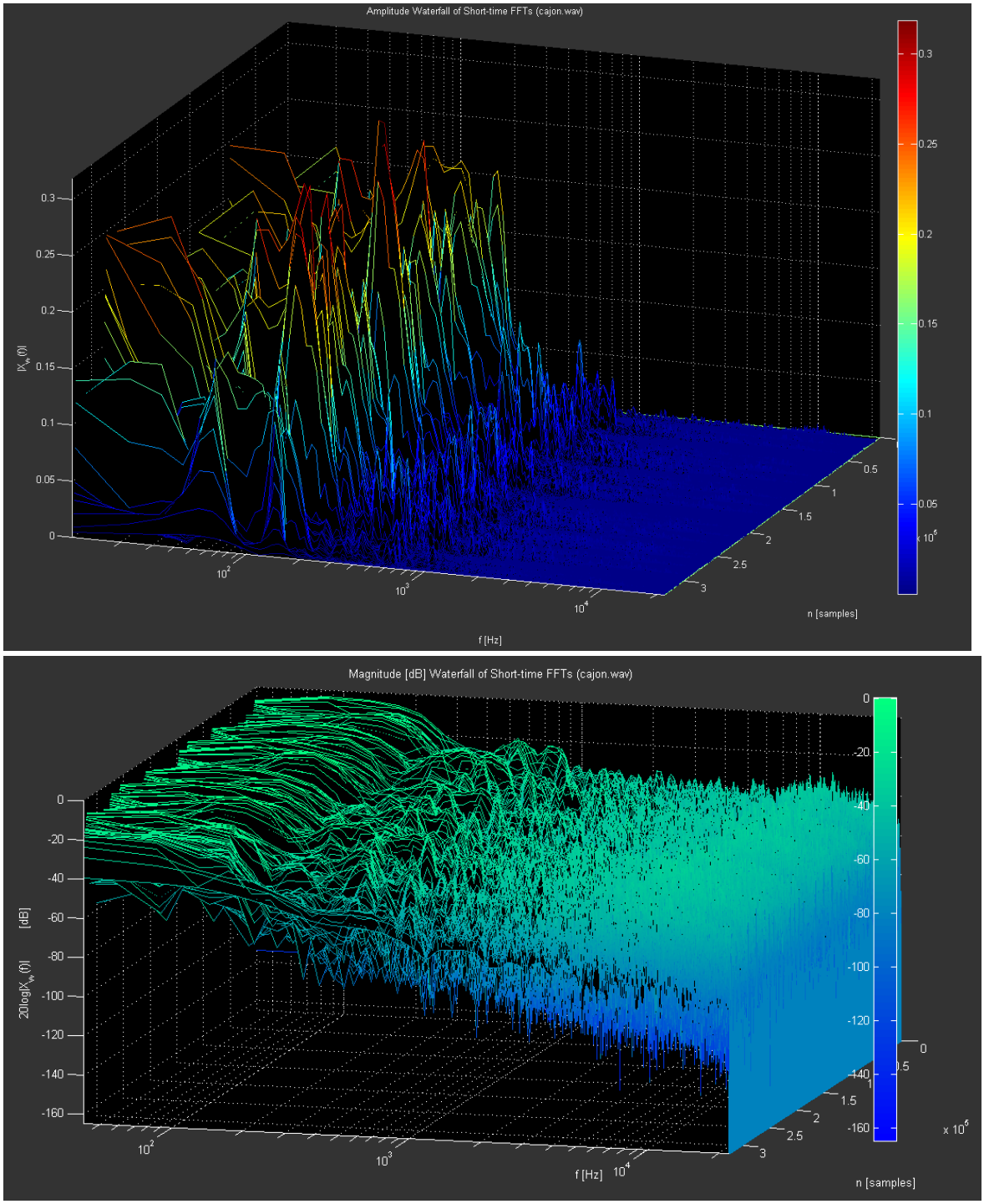

Figure 27: Time frequency representation, Cajon percussion dry amplitude (top) and magnitude (bottom) waterfalls with size an FFT size of 4096 samples.

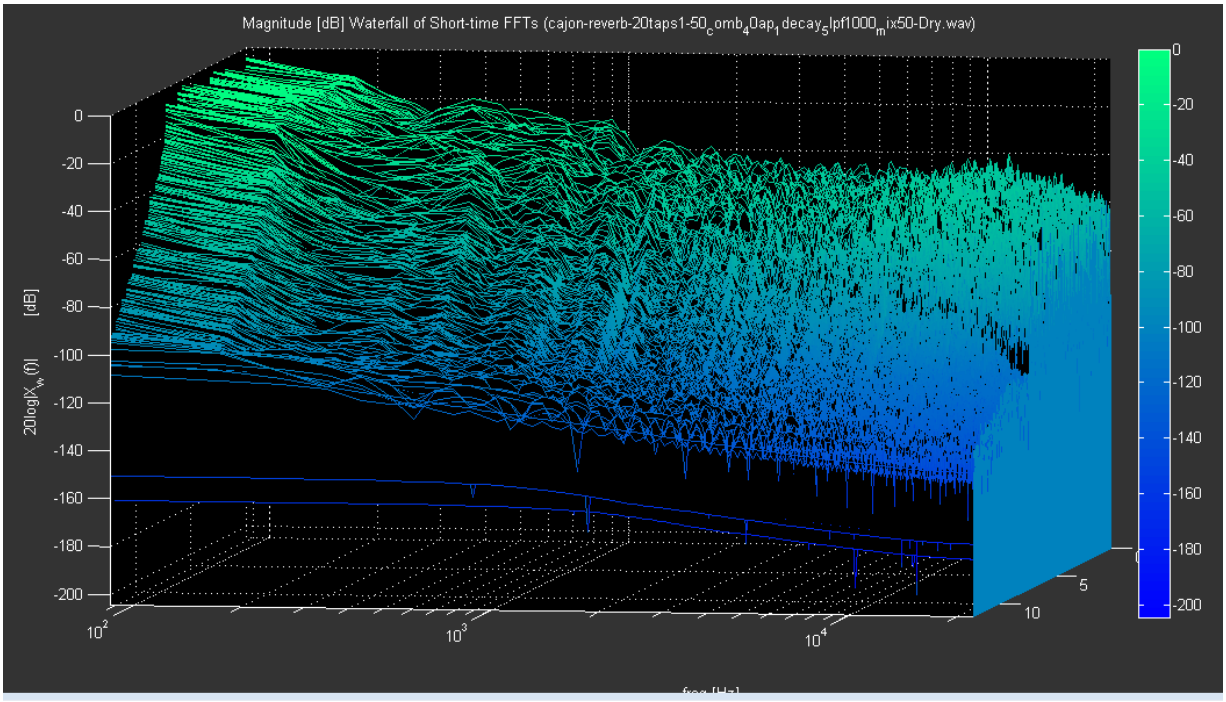

Figure 28: Time frequency representation of the Cajon percussion clip after being processed with the Moorer's reverberator. Output audio file is cajon-reverb-20taps1-50_comb_40ap_1decay_10lpf1000_mix50.wav. The low end response is enhanced through lowpass filtering.

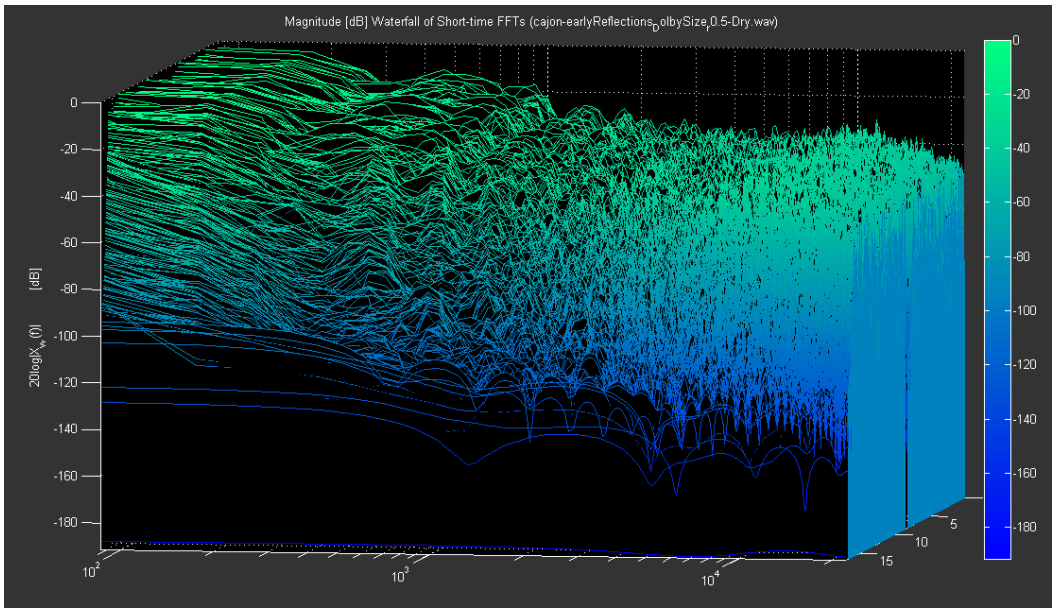

Figure 29: Time frequency representation of the Cajon percussion clip after being processed with early_reflections.m (purely convolved with a basic simulated room IR) Output audio file is earlyReflections_DolbySize_r0.5.wav.

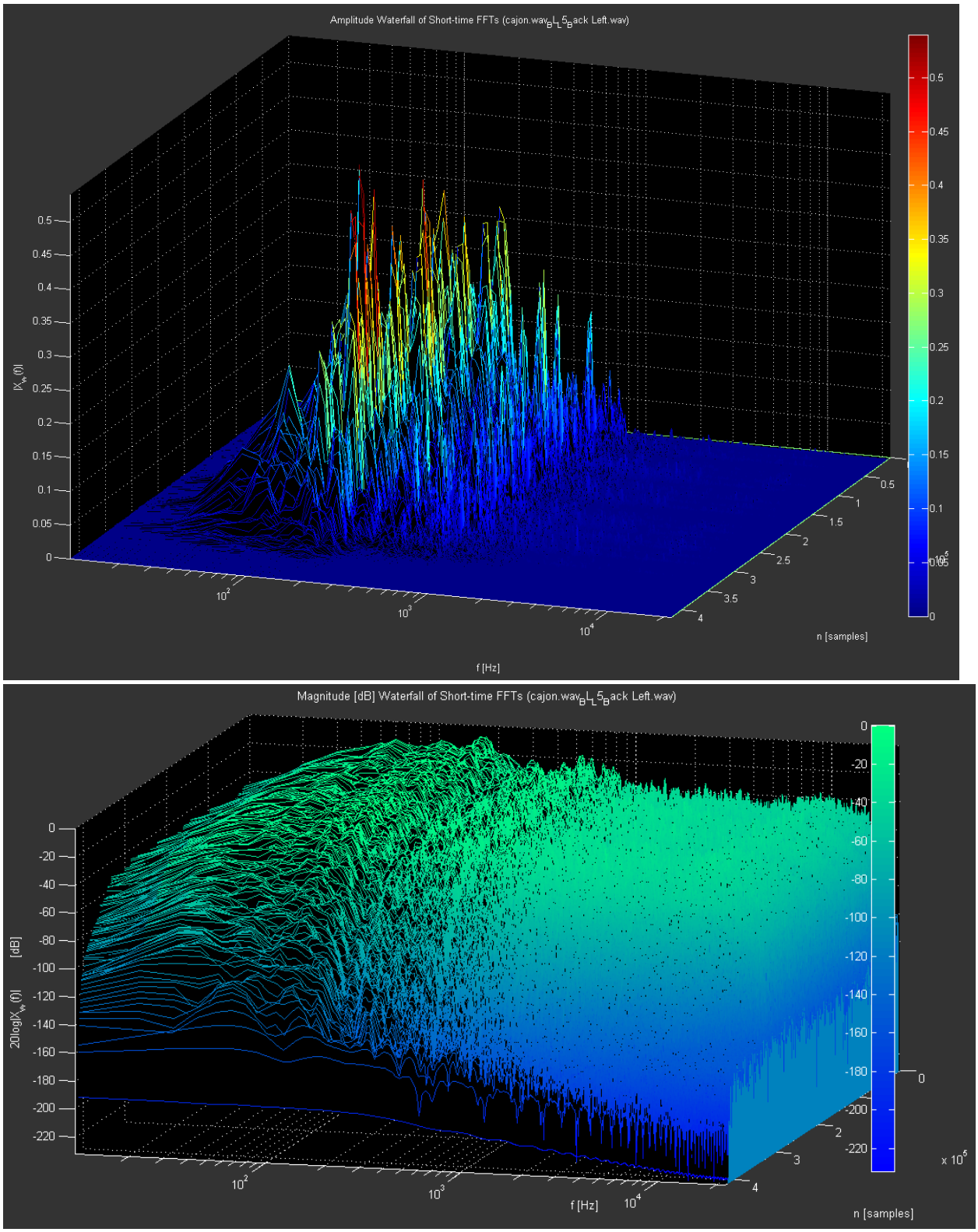

Figure 30: Cajon percussion clip convolved with a RIR that was measured in the Philip T Young Recital Hall at the University of Victoria. Response below 30 Hz is not well represented due to the selected measurement range (20 – 22 kHz) and the poor low frequency response of the speakers.

6. CONCLUSIONS

Using the hybrid reverb concept outlined, a reasonably realistic reverb can potentially be achieved in a way that is less computationally intensive than pure convolution reverb (convolution of a dry signal with the measured the IR of a real room, or a high order FIR filter). High computational efficiency was not the priority for this prototype, however. The intention was to employ an algorithmic reverb model with reasonably a high level of complexity, and stack it up against convolution reverb with measured room impulse responses. While the results of this particular reverb are not comparable to a physical acoustic environment, the design concept can be expanded to use more advanced room simulations.

This was an excellent learning experience, as designing a reverb algorithm and choosing parameterization to achieve a desired outcome is challenging. While we did not create high quality reverb, we did generate reasonable algorithmic reverbs that can be parameterized in many ways. With the Moorer's reverberator, we were able to achieve a sound that resembles a large, reverberant hall. Using the hybrid reverberator that incorporated a simulated room IR for early reflection modeling, a small room was modeled quire well. A fundamental requirement for implementing the hybrid model is synchronization between the two main stages (early reflections and late reverb). The underlying framework of the two stage reverberator is robust: various algorithms for early reflections and late reverberation can be implemented together using this concept. However, the continuity between the reverb stages that leads to a consistent spatial impression was found to be inferior to convolution reverb (in its pure form) and modern feedback delay matrix reverb algorithms.

7. REFERENCES

CITED REFERENCES

[1] Beranek, L., " Concert Hall Acoustics—2008*," J. Audio Eng. Soc., vol. 56, no. 7/8, pp. 532-544, 2008 July/August.

[2] Sound on Sound, Online Image, 2006 [2009 July 26], Available at FTP: http://www.soundonsound.com/sos/may00/articles/reverb.htm

[3] Udo Zölzer, *DAFX*., (John Wiley & Sons, 2002., West Sussex)

[4] Toma, N.; Topa, M.D.; Popescu, V.; Szopos, E., "Comparative Performance Analysis of Artificial Reverberation Algorithms," *Automation, Quality and Testing, Robotics, 2006 IEEE International Conference on* , vol.1, no., pp.138-142, 25-28 May 2006
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4022835&isnumber=4022803

[5] Toma, N.; Topa, M.; Szopos, E., "Aspects of reverberation algorithms," *Signals, Circuits and Systems, 2005. ISSCS 2005. International Symposium on* , vol.2, no., pp. 577-580 Vol. 2, 14-15 July 2005
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1511306&isnumber=32298

[6] Campbell, D. (06 June 2007). RoomSim acoustic toolbox for MatLab.

[7] McGovern, Stephen G. A Model for Room Acoustics, 2004

[8] Smith, W. (2007), *The Scientist and Engineer's Guide to Digitial Signal Processing*, [On-line], Available at FTP: http://www.dspguide.com/pdfbook.htm [2009, July 27].

[9] Smith, J. (2009, Mar.), *Spectral Audio Signal Processing*, [On-line], Available at FTP: http://ccrma.stanford.edu/~jos/sasp/ [2009, July 27].

GENERAL REFERENCES

Rocchesso, D.; Smith, J.O., "Circulant and elliptic feedback delay networks for artificial reverberation," *Speech and Audio Processing, IEEE Transactions on* , vol.5, no.1, pp.51-63, Jan 1997
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=554269&isnumber=12065

P. R. Newell, Recording Studio Design, 2 ed. , New York: Focal Press, 2007.

APPENDIX A – Selected MATLAB Scripts

Additional Scripts are Contained in the Project Folder

```
%========================================================================
% reverbHybrid.m     Authors: Hundson Giesbrecht, Will Mcfarland, Tim Perry
% Elec 407
% DSP Project                                                    2009-07-21
%
% Implements a multi stage hybrid algoritmic reverberator.
% Stage A: Early Reflections
%   -Option 1: Tap delay line
%   -Option 2: Simulated room IR reflection modeling by convolution with
%              low order FIR filter.
% Stage B: IIR comb, lowpass & allpass filters
%   -IIR based algorithm for late reverb using comb, lowpass, and
%    allpass filters
%
% y=REVERBHYBRID(x,sr,ntaps,tapminms,tapmaxms,ncomb,nap,Td,lpf,mix,compRIR)
%
%parameters
% x         -input signal
% sr        -input signal's sampling rate
% ncomb     -number of comb filters
% nap       -number of all-pass filters
% Td        -time for decay in sec
% lpf       -feedback LP filter passband frequency (0 for no lowpass
%            filtering) to simulate high frequency rolloff
% hpf       -input frequency filtering
% mix       -dry/wet mix percentage (0 - 100)
% y         -output signal
% compRIR   -0 to use tap delay line early reflections, 1 to use room IR

function
y=reverbHybrid(x,sr,ntaps,tapminms,tapmaxms,ncomb,nap,Td,lpf,mix,compRIR)

    %----------------Initializations-------------------
    %make input and output signals a bit longer so that reverb tail can be
    %heard
%   y=zeros(1,length(x));
%   y2=zeros(1,length(x));
    y=zeros(1,length(x)+(Td)*sr);
    apOuts=zeros(1,length(x)+(Td)*sr);    % comb-allpass portion y
    x0 = x;
    x=[x;zeros((Td)*sr,1)];


    %HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
    % EARLY REFLECTIONS - tap delay or simulated room IR
    %========================================================================


        if compRIR == 0
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % EARLY REFLECTIONS METHOD 1 - tapped delay line
            % calculate delay tap locations IN SAMPLES
```

```
            %
            h = 0;
            minms = tapminms;
            maxms = tapmaxms;
            mindelay = ceil(minms / 1000 * sr);
            maxdelay = ceil(maxms / 1000 * sr);
            tapsamplevalues = ceil(rand(ntaps,1).*(maxdelay-
mindelay)+mindelay);
            tapsamplevalues = sort(tapsamplevalues);
            tapdelaylen = max(tapsamplevalues);
            tapdelay = zeros(tapdelaylen,1);%create the delay line
            tapgains = zeros(1,ntaps);

            %set the gains of the taps to decrease from 1 to 0 linearly
            for a=1:ntaps
                tapgains(a) = ((a - ntaps) * -1 / ntaps)*.5;
            end

            %plot these values
            plotvals = zeros(1,tapdelaylen);
            for a=1:ntaps
               plotvals(tapsamplevalues(a)) = tapgains(a);
            end
            plot(plotvals);


        else    %getting the early reflections from geometric modeling
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % EARLY REFLECTIONS METHOD 2 - Simulated Eoom IR
            % FFT convolve input with FIR filter for modeled room
            %
             mixRIR = 100;

            disp('Early Reflections modeling with RIR convolution...')
            %[earlyVerb]=early_reflections(x,mixRIR,sr,micpos,n,r,rm,src)
            [earlyVerb, hl, hr] = early_reflections(x0,mixRIR,sr);
            h = hl;

            %-----Make compatable with stage B (late reverb)---
            tic
            disp('Simulated Room IR Reflection Post-Processing...')

            tapdelaylen = length(h);
            tapdelay = zeros(tapdelaylen,1);

            threshcount = 0;
            for a=1:length(h)
                if (h(a) > 0.01)
                    threshcount = threshcount + 1;
                end
            end

            ntaps = threshcount;
            tapsamplevalues = zeros(threshcount,1);
            tapgains = zeros(threshcount,1);
            count = 0;
            for a=1:length(h)
```

```
            if (h(a) > 0.001)
                count = count + 1;
                tapsamplevalues(count) = a;
                tapgains(count) = h(a);
            end
        end
    end
    toc




%HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
% REVERB TAIL - IIR comb filters
%  calculate comb filter delay amounts (in samples)
%  should be in the range of 50 to 80 ms (Moorer's preferred)

if length(h) > 1
    startsample = tapsamplevalues(threshcount);
else
    startsample = maxdelay;
end
%new max delay is the startsample + 30 ms (calculated in samples)
maxdelay = ceil( startsample + (50/1000 * 44100));

%get the max delays for each comb filter
% by finding N coprime samples, from the max tap ms to another value (+30
ms)
% the first coprime value is STARTSAMPLE

%get the max delays for each comb filter, all coprime

combdelayvalues = coprimes(ncomb,startsample,maxdelay);
%combdelayvalues = [2205 2600 2800 3000 3100 3200]; % NOT COPRIME
%EXAMPLE


%gains for each comb filter
combgains = 10.^(-3*Td*sr./combdelayvalues);
%create the delay lines
combdelaylen = max(combdelayvalues);
combdelays = zeros(ncomb,combdelaylen);

%todo: replace with t60
for a=1:ncomb
   combgains(a) = 10^(-3*(combdelayvalues(a)/44100)/Td);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALLPASS filter to increase dispersion
%
g=.6; %parameter for ap filter
delayval = 6;
```

```
    apdelaymax = ceil(delayval / 1000 * sr); %6 ms delay in each all-pass
filter
    apdelays = zeros(nap,apdelaymax);
    apfbdelays = zeros(nap,apdelaymax);

    ptr = 1;    %pointer to delay line

    disp('Late Reverb Processing with IIR Comb-Allpass Kernal Algorithm...')
    tic

    for k=1:length(x);

        if (mod(k,44100) == 0)
            k;
            length(x);
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % tapped delay line
        tapsum = 0;
        %ptr = current position in delay line
        for a=1:ntaps
            tapsum = tapsum + tapgains(a)*tapdelay(mod(ptr +
tapsamplevalues(a),tapdelaylen-1)+1);
        end
        %write the new value in the delay line
        tapdelay(mod(ptr,tapdelaylen-1)+1) = x(k);


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %pass into the parallel comb filters
        combsum = 0;

        for a=1:ncomb
            combout = tapsum +
combgains(a)*combdelays(a,mod(ptr+combdelayvalues(a),combdelaylen-1)+1);
            combsum = combsum + combout;
            %update the delay line
            combdelays(a,mod(ptr,combdelaylen-1)+1) = combout;
        end
        apin = combsum/ncomb;

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         %pass this into a series of all-pass filters
        for a=1:nap
            % diff eq'n: y(n) = g*x(n) + x(n-m) + -g*y(n-m) where m is
            apout = g*apin + apdelays(a,mod(ptr + apdelaymax,apdelaymax-1)+1)
+ -g*apfbdelays(a,mod(ptr + apdelaymax,apdelaymax-1)+1);
            %they are connected in series, so here we go
            apin = apout;
            %update the delay lines
            apfbdelays(a,mod(ptr,apdelaymax-1)+1) = apout;
            apdelays(a,mod(ptr,apdelaymax-1)+1) = combsum;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        y(k)= tapsum + apout;

        apOuts(k) = apout;

       %increment the pointer for the delay lines
       ptr = ptr + 1;
    end
    toc

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Adjust late reverb timing based on initial time delay gap
    if compRIR == 1
        earlyVerbzp = [earlyVerb zeros(1,length(y)-length(earlyVerb))]; %
zero pad
        [earlyMAX eMAXind] = max(abs(earlyVerbzp));
        [lateMAX lMAXind] = max(abs(apOuts));

        %---------------------------------------
        % find ITDG of IR
        %---------------------------------------
        [maxRIR, startIR] = max(abs(h));
        h_temp = h;
        h_temp(startIR) = 0;
        [maxReflec, ITDind] = max(h_temp);

        ITDG = startIR - ITDind;        % initial time delay gap in samples

        DelayComp = eMAXind - lMAXind - 2*ITDG;  %late verb timing adjustment

        apOutsComp = [apOuts(DelayComp:length(apOuts))];
        apOuts = zeros(1, length(apOuts));
        apOuts(1:length(apOutsComp)) = apOutsComp;

        y = earlyVerbzp + apOuts*earlyMAX/lateMAX/8;  % scaled output with
timing correction
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % LOWPASS filter to simulate air absorption of high frequencies
    [lpb,lpa] = butter(1,lpf/sr);
    y = filter(lpb,lpa,y);

    %actual output mix
    y=y./max(y);
    ny = (mix/100)*y + ((100-mix)/100)*x';
    y= ny;
    y=y./max(y);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% early_reflections.m                                          Authors:
%                                                            Hundson Giesbrecht
% Elec407                                                    Will Mcfarland
% DSP Project                                                TimPerry
% Elec407                                                    2009-07-24
%
% Models the early reflections of reverb based on a simulated room
% impulse response model. Designed to be used as the FIR stage of a
% FIR/IIR reverb algorithm (where the IIR portion computes the late
% reverberation using a Schroeder reverberator or similar algorithm).
%
%  [EARLYVERB]...
%       EARLY_REFLECTIONS(input,hop_size,mix,fs,micpos,n,r,rm,src)
%
%           x           `-input vector
%           mix          -dry/wet mix percentage (0 - 100)
%           fs           -sampling frequency
%           micpos       -mic position vector [xm ym zm] in the room
%           n            -number if reflections to track (if n=6, echos
%                        that have been reflected <= 6 times off of a wall
%                        will be computed.
%           r            -reflection coefficient (typically -1<r<1) where
%                        r is related to the absorbtion coefficient
%                        alpha by the relation: alpha = 1 - r^2
%           rm           -room width, height, depth [ rw rh rd ]
%           src          -source position vector [xs ys zs] in the room
%
%
%           earlyverb    -output vector of convolved signal
%
% REFERENCES:
% [1] Udo Zölzer, DAFX. John Wiley & Sons, 2002.
% [2] Stephen G. McGovern, Room Impulse Response function RIR, 2003
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%function [earlyVerb]=early_reflections(input,mix,fs,micpos,n,r,rm,src)
function [earlyVerb, hl, hr]= early_reflections(input,mix,fs);

clear all;
close all;
wavfile='cajon';
[input fs nbits] = wavread(wavfile);

%wet=mix; %percentage of the signal that is processed with reverb

% ============================================================
% Compute room IR model for early reflections
% ============================================================

% Define room dimentions for generation of room impulse response
% Example values that use Dolby's optimum ratio for film and music rooms
% are   Mean Height = 12', Mean Width = 17.91', Mean Length = 27.76'
rw = 17.91;                 % room width
rh = 13.5;                  % room height
rd = 27.76;                 % room depth
```

```
rm = [rw rh rd ];

src = [ rw/2 rh/3 rd/6 ];   % source position in the room
src = [ rw/7 rh/3 rd/6 ];   % test imaging with source placed not central

% Stereo microphone configuration (typical spaced omni pair is modeled)
micposL = [ (rw/2 - 1.5) rh/2 rd/3 ];   % position of left mic in the room
micposR = [ (rw/2 + 1.5) rh/2 rd/3 ];   % position of right mic in the room

% n, the number of reflections to track (ex. n=6 means echoes that have
% undergone up to 6 reflections will be accounted for)
n = 10;
% r, reflectivity coefficient of the walls/floor/cieling
r = 0.5; % 0.3 is roughly a plywood sheet

    %-------------get IRs for room--------------
hl = rir(fs, micposL, n, r, rm, src);
hr = rir(fs, micposR, n, r, rm, src);

    %-----------for stereo IR simulation---------
IRL = hl;
IRR = hr;



% ============================================================
% Initializations for convolution with modeled room IR
% ============================================================

dry = input;            % unprocessed signal
input_size=size(dry);

%check if the input is stereo and separate the vector
if input_size(2)==1
   x1l=dry; %set variable x1 to the input signal
   x1r=x1l;
else
    x1l=dry(:,1);
    x1r=dry(:,2);
    is_stereo=1;
end

    %---Convolve with room IR using High Speed Convolution----
yl = speedConv(dry, IRL);
yr = speedConv(dry, IRR);



% ============================================================
% Scaling Output Levels
% ============================================================
yrsc=yr; %keep another copy for scaling
ylsc=yl;

if max(yl)>1 || max(yr)>1
%Rescale
if max(yl)>max(yr)
```

```
yl=0.98*ylsc*(max(abs(x1l))/max(abs(ylsc)));
yr=0.98*yrsc*(max(abs(x1l))/max(abs(ylsc)));
display('left is bigger')
else
yl=0.98*ylsc*(max(abs(x1r))/max(abs(yrsc)));
yr=0.98*yrsc*(max(abs(x1r))/max(abs(yrsc)));
display('right is bigger')
end
end


yst=[yl yr];
x11=[x1l;zeros(length(yst)-length(x1l),1)]; %double the mono input so it can
be blended with the stereo output




%HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
% Early reflections output & plots

%================================================
% for mono output
earlyVerbMon = yl;             % mono output (left channel)
%yl = 0.98*yl/max(abs(yl));
earlyVerb = yl;

% %================================================
% % for stereo output
% output=yst; %wet/100*yst+(100-wet)/100*x11;
% x11=[x11 x11];
% earlyVerb = yst;



% write eary reflection portion of reverb for testing
mkdir(pwd,'Processed Audio');
wavename=strcat(wavfile,'-earlyReflections','.wav');
dirwavename=fullfile(pwd ,wavename);
%wavwrite(yl,fs,nbits,dirwavename);
wavwrite(yst,fs,nbits,dirwavename);



%---------------plot room IR-----------------------

rirFIG = figure('Name','RIR Early Reflections', ...
    'Position',[100,100,700,450]);
figNumStart = get(0,'CurrentFigure');
clear figure(figNumStart);
figNum = figNumStart + 1;


maxAmpIR = max([max(abs(hl)) max(abs(hr))]);
axis_IR = [0, length(hl), 0, 1.2*maxAmpIR];

subplot(2,1,1)
```

```
stem(hl,'b.');
axis(axis_IR);
grid on;
title(['Simulated Room IR hl[n] (Left Channel) for Early Reflection
Modeling']);
ylabel('h[n]')
xlabel('n (samples)')
subplot(2,1,2)
stem(hr,'r.');
axis(axis_IR);
grid on;
title(['Simulated Room IR hr[n] (Right Channel)  for Early Reflection
Modeling']);
ylabel('hr[n]')
xlabel('n (samples)')



%---------------plot convolution result------------
convFIG = figure('Colormap',jet(128),'Name','Conv with RIR', ...
    'Position',[100,100,1000,300]);
figNum = figNum + 1;

t=(1:length(earlyVerbMon))/fs;
plot(t,earlyVerbMon);
title('Input Convolved with Simulated RIR (Early Reflections)')
xlabel('Time (Seconds)')
ylabel('Amplitude')
```

```
%==========================================================================
% SpeedConvOLA.m                                          Author: Tim Perry
% Elec407: DSP                                             V00213455
% Hybrid Algorithmic Reverb Project                       2009-07-24
%
% High Speed Convolution
% Convolves two sound files using Acyclic (aperiodic) FFT convolution in
% frequency domain.Zero padding, rectangular windows overlap-add are used.
% Zero padded acyclic convolutions are imbedded in cyclic convolutions to
% optimize computation speed, but prevent time domain aliasing caused by
% cyclic convolution.
%   [y] = SPEEDCONV(input, filter)
%
%             y = output vector of convolved signals
%             x = input vector to convolve with filter
%             h = filter for FFT filtering (Ex: simulated room IR)
%
%==========================================================================

function [y] = speedConv(x, h)
% function [y] = speedConv(input, filter)
% [h fs nbits] = wavread(filter);
% [x fs nbits] = wavread(input);



%-----------------------------------------------
% Alternative Inputs (2 audio files)
%-----------------------------------------------
% filter = 'dBu_IR L1-C_PTY_Center_74.wav'
% input = 'cajon.wav';
% [h fs nbits] = wavread(filter);
% [x fs nbits] = wavread(input);

% % Default to left channel for mono treatment of stereo input files
% x = x(:,1);                % (since left channel is col vectors)
% h = h(:,1);

Nx = length(x);                % original length of audio files
Nh = length(h);

%----------------check input files------------------------------
% If col vectors, change to row vectors for matrix operations
if size(h,1)>1,
  h=h';
end
if size(x,1)>1,
  x=x';
end


%--------------------------------------------------------------------------
% Parameters for Acyclic conv with rectangular windows and overlap-add
%--------------------------------------------------------------------------
N = Nx;                              % Set Nx to be the signal length
L = Nh;                              % Set Nh to be the filter length
M = 4096;

    %--------Option 1 FFT Size----------
%NFFT = 2^nextpow2(M + Nh + - 1);
```

```
    %------Option 2 FFT Size (Faster in General)-------
NFFT = Nh+M;                                % size of FFT
M = NFFT-Nh+1;                              % optimized window length

Ra = M;                           % hop size for acyclic with rectangular win
Nframes = 1+floor((N-M)/Ra);           % # of full blocks to convolve
% NsigOut = 2^nextpow2(N + NFFT)       % for Acyclic, NsigOut is smallest
%                                      % power of 2 that is  >= |Nx+Nh-1|


tic
%HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
%============================================================
% Acyclic Convolution (Frequency Domain) with zero padding
%============================================================
hzp = [h zeros(1,NFFT-Nh)];         % pad filter kernal h to FFT size
H = fft(hzp);                       % frequency response of filter kernal

y = zeros(1,N + NFFT);                    % vector for output signal

%---apply FFT convolution with overlap add----
for m = 0:(Nframes-1)
    index = m*Ra+1:min(m*Ra+M,N);         % mth frame indices
    xm = x(index);                        % rectangular windowed mth frame
    xmzp = [xm zeros(1,NFFT-length(xm))]; % zero padding

        %----Frequency domain processing----------
    Xm = fft(xmzp);         % FFT of padded mth frame
    Ym = Xm .* H;           % freq domain multiplication = time domain conv

        %----Synthesis (overlap-add)-----------
    ym = real(ifft(Ym));                % return results to time domain
    indexOut = m*Ra+1:(m*Ra + NFFT);
    y(indexOut) = y(indexOut) + ym;     % apply overlap adding
end

y = y(1:Nx + Nh - 1);        % truncate output signal to length Nx + Nh + M

%HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
toc


%-----------------------------------------------------
% Output Audio File
%-----------------------------------------------------
y = y/max(abs(y));  %normalize output mag to prevent wavwrite  clipping
y = y*0.98;           % bring level down further

% file_out=strcat(input(1:length(wavfile2)-4),'-SpeedConvOLA-',wavfile1(1:8),'.wav');
% name output file
% wavwrite(y, fs, nbits, file_out);                % write output audio file
%
%
% wavplay(y, fs);                                  % play processed output



%-------------------------------------------------------------
% Plots (Time Domain)
%-------------------------------------------------------------
maxAMP = max([max(abs(hzp)) max(abs(x)) max(abs(y))]);
NsigOut = length(y);
N = Nh -1;
```

```
Lwin = 2*(Nh - 1);


Nsamples_h = linspace(0,Nh,Nh);
Nsamples_x = linspace(0,Nx,Nx);
Nsamples_padded = linspace(0, length(hzp), length(hzp));
Nsamples_xm = linspace(0, length(xm), length(xm));
Nsamples_Out = linspace(0,NsigOut,NsigOut);

axis_block = [0 Lwin -1.1*maxAMP 1.1*maxAMP];
axis_inputs = [0 NsigOut -1.1*maxAMP 1.1*maxAMP];
axis_result = [0 NsigOut -1.1*maxAMP 1.1*maxAMP];
% axis_inputs = [0 Lwin -1.1*maxAMP 1.1*maxAMP];
% axis_result = [0 Lwin -1.1*maxAMP 1.1*maxAMP];

figure (2)
% ---------Freq Domain Convolution results plotted in time domain-----
subplot(4,1,1);
plot(Nsamples_x, x/max(abs(x)),'k-');
axis(axis_inputs)
grid on;
title('x[n]   (dry input signal)')
ylabel('x[n]')

subplot(4,1.3,2.45);
plot(Nsamples_padded, xmzp/max(abs(x)),'k-');
axis(axis_block)
grid on;
title(['x_izp[n]     (windowed, zero padded block of x[n]), i=',num2str(m)])
ylabel('x_izp[n]')

subplot(4,1.3,3.75);
plot(Nsamples_h, h/max(abs(h)),'r-');
axis(axis_block)
grid on;
title('hzp[n]   (zero padded filter [simulated room IR])')
ylabel('h[n]')

subplot(4,1,4);
plot(Nsamples_Out, y,'b-');
axis(axis_result)
grid on;
title('y[n] = h[n]*x[n] (Ayclic FFT Convolution with Overlap Add)')
xlabel('n')
ylabel('y[n]')
```