Modifications to Multiverb

Warren L. G. Koontz

June 8, 2015

1 Introduction

Multiverb is an algorithm to add a reverberation effect to an audio signal. The algorithm is based on a network of multi-port acoustic elements and is described in detail in [1]. I have implemented the algorithm as a MATLAB function, a MATLAB "testbench" and as a VST plug-in for Windows applications. The original VST implementation allows the user to vary the following parameters:

- The low- and high-frequency reverberation times. These help to determine the rate of decay of the reverberations.
- Separate gain controls for the dry (unprocessed) and wet (processed) components of the audio output.
- The degree of separation of the left and right stereo channels.

As described in the next section, the new implementation provides the user with an additional parameter to vary the reverberation effect.

2 Description of Changes

The Multiverb network consists of a pair of identical multi-port acoustic junctions connected by parallel acoustic waveguides of different lengths as illustrated in Figure 1. Each acoustic junction is characterized by a scattering matrix $\mathbf{S}$ whose element values depend only on the number of ports. The parallel waveguides are characterized by a (diagonal) matrix $\mathbf{H}(z)$ of transfer functions that introduce sample delays and frequency-dependent attenuation. In the original implementation, the sample delays are fixed at compile time. This creates two limitations:

- Although varying the sample delays will change the effect, the user cannot vary the delays.

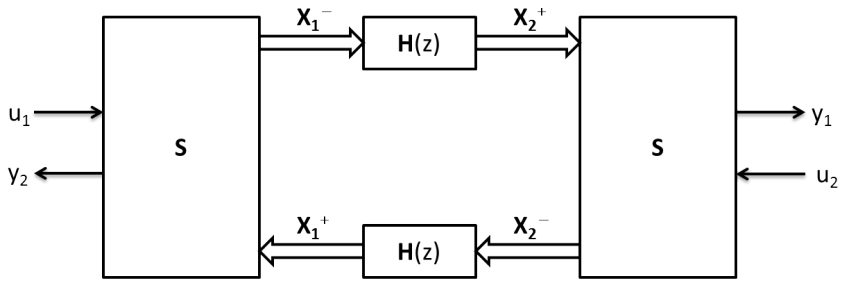

Figure 1: Multiverb Model

- The fixed delays are based on a sampling rate of 44100 Hz and may not be suitable for other rates.

Thus my new implementation gives the user some control over the sample delays.

2.1 Dynamic Waveguide Delay Control

In Multiverb a waveguide is implemented as a buffer whose length is equal to the sample delay. The number of waveguides N is (still) fixed at $N = 8$. The method that I have been using to determine the N sample delays is as follows:

1. Start with some maximum sample delay $D_{\max}$ (say $D_{\max} = 5000$).
2. Calculate a related minimum sample delay $D_{\min}$ (say $D_{\min} = D_{\max}/10$).
3. Determine an exponentially decreasing sequence of N delays starting with $D_{\max}$ and ending with $D_{\min}$.
4. Adjust these delays to a nearby prime number.

Thus the delay values make up a sequence of prime numbers that approximates an exponential decrease from $D_{\max}$ to $D_{\min}$.

Why prime numbers? A journal reviewer asked me the same question and was not happy with my answer! Nevertheless, I do this because Multiverb's impulse response includes echo impulses delayed by a huge number of combinations of these delay times and I want to reduce the number of echo impulses that have the same delay.

Here is the MATLAB version of the algorithm:

```
function d=mkDelays(N,maxDelay)
minDelay=maxDelay/N;
alpha=(minDelay/maxDelay)^(1/(N-1));
d=zeros(N,1);
for n=1:N
```

```
    d1=round(maxDelay*alpha^(n-1));
    while (isprime(d1)==false)
        d1=d1-1;
    end
    d(n)=d1;
end
end
```

Multiverb now includes a C++ version of this algorithm that is invoked as part of the `update` method that is called whenever the user tweaks a slider on the GUI. And the GUI now has a slider that determines $D_{\max}$.

Since "Maximum Waveguide Sample Delay" may not be meaningful to many users, I have decided to offer the user a parameter called "Room Area" measured in square feet (range: 1000 to 10000). Multiverb uses some voodoo mathematics to convert room area to $D_{\max}$. Here's the voodoo rationale:

- The length of the longest waveguide is the length of the room.
- The the floor of the room is a golden rectangle.

The length and width of a golden rectangle are related by

$$L = \phi W \tag{1}$$

where

$$\phi = \frac{1+\sqrt{5}}{2} \tag{2}$$

From this it follows that the length of the room can be determined from the area as

$$L = \sqrt{A\phi} \tag{3}$$

The time required for sound to travel a distance L is $T = L/c$ where c is the speed of sound (1116.43701 feet/second at sea level). The corresponding sample delay is $D_{\max} = f_s T$ so that $D_{\max}$ is given by

$$\begin{aligned} D_{\max} &= f_s L/c \\ &= f_s \sqrt{A\phi}/c \\ &= \gamma f_s \sqrt{A} \end{aligned} \tag{4}$$

where

$$\gamma = \sqrt{\phi}/c \tag{5}$$

At sea level $\gamma = 0.0011$ so that our final voodoo formula is

$$D_{\max} = 0.0011 f_s \sqrt{A} \tag{6}$$

where f_s is the sampling rate in Hz and A is the room area in square feet. Table 1 lists the maximum delays for a few combinations of room area and sampling rate. On the basis of this data, I have sized the waveguide buffers for to allow up to 24000 samples. The computed delays, all of which should be less than this maximum, determine when the buffer pointers are reset.

Area (sq ft)	Sampling Frequency (Hz)	Maximum Sample Delay
1000	44100	1534
1000	192000	6679
10000	44100	4851
10000	192000	21120

Table 1: Sample Delays For Various Room Areas And Sampling Rates

When the room area slider, and therefore all of the waveguide buffer lengths, are varied as the audio is being played or recorded, anomalies can occur. For example, suppose the system has been running for some time and the user increases the room area. Then all of the delays increase and buffer space that was previously unused becomes used. I have designed the code so that unused buffer space is always set to 0. Therefore a string of zeros is inserted into each waveguide. Similarly, if the user decreases the room size, some samples in the buffers are discarded.

2.2 Other Changes

The original implementation of Multiverb provides two gain controls for the user: Dry Gain and Wet Gain. These two controls allow the user to vary the mix of the processed signal with the original to vary the degree of reverberation. I have replaced these with a Master Gain control (gain in dB) and a Reverb Mix control (0 to 100%). Nothing new, just a different approach.

References

[1] Koontz, W.L.G., Kim, S-Y., and Indelicato, M.J., "A Digital Reverberation Simulator Based On Multi-Port Acoustic Elements", *JMEST*, Vol. 2, No. 1, January, 2015.