3D Convolution Reverberation Synthesis

Olivia Canavan, Sam Schachter, Kyle Tworek

University of Rochester Audio and Music Engineering

HAJIM SCHOOL OF ENGINEERING & APPLIED SCIENCES
UNIVERSITY of ROCHESTER

Introduction

Creating multidimensional reverbs using measured impulse responses is challenging and unrealistic for most users, as it requires an ambisonic microphone array. Because of this, we have designed a method for calculating and implementing a multidimensional reverb using a synthesized impulse response, customized based on user input. This method has produced realistic reverberated output in a 25.2 speaker array.

Image 1. Playback environment, 25.2 speaker system.

Graphical User Interface

In an effort to create a highly personalized user experience, we designed a GUI that allows the user to select specific aspects of the desired room. The user is able to specify the dimensions of the room, one of several preset room types which correspond to its reflectivity, and the location of the sound source relative to the receiver.

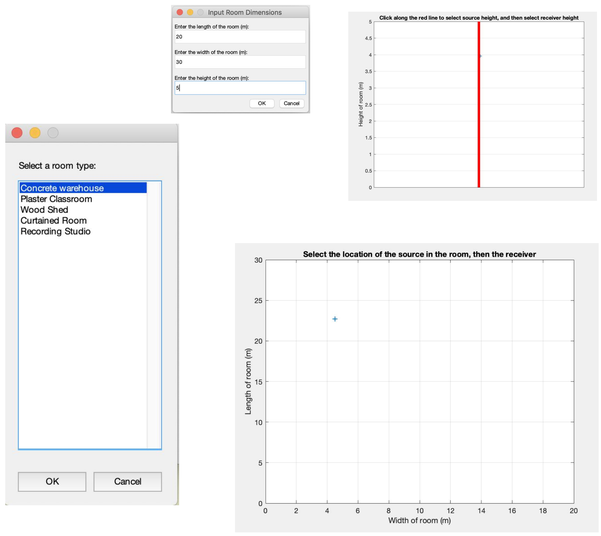

Fig 1. Examples of our graphical user interface.

Multidimensional Impulse Response

Given the room dimensions and reflectivity obtained from the GUI, we are able to calculate the multidimensional impulse response of the room by the method of images (Figure 3). For a set number of images, we are able to simulate the number of reflections necessary for each reverberated impulse to reach the receiver, the magnitude of each impulse, the time it arrives, and the direction from which it arrives. Then, by scaling each impulse according to the room's reflectivity, we achieve an accurate representation of the 3-dimensional impulse response of the room [1].

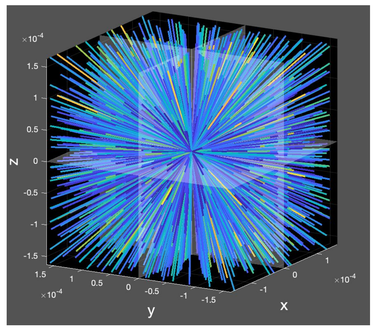

Figure 2. Graphic visualization of the multidimensional impulse response generated by our program using user specifications.

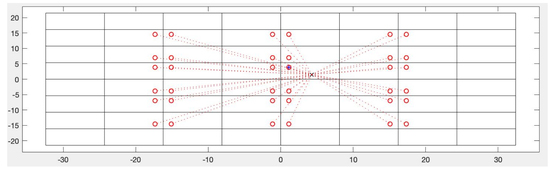

Figure 3. 2-dimensional representation of method of images calculation. Each dotted line represents a vector from the receiver to an image of the source. Each room and source image represents a reflection off of the adjacent wall in the real room. The number of walls a vector crosses through before it reaches the receiver represents the number of walls the sound must reflect off of in the real room before it reaches the receiver [2].

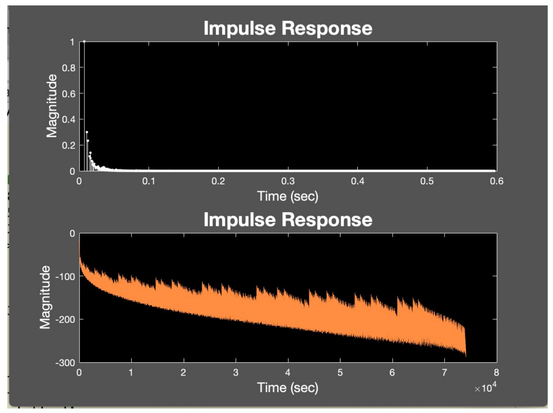

Figure 4. The normalized and log-amplitude magnitudes of the generated multidimensional impulse response

Vector Base Amplitude Panning

The raw 3-D impulse response generated in the previous stage of the program contains position data for each individual 'spike' across the time domain of the response, as visualized in Figure 2. In order to create a 'virtual source' at each of these many unique points in 3-D space using a finite number of (in our case, 25) speakers, we implemented the Vector Base Amplitude Panning (VBAP) method as described by Ville Pulkki in his 1997 paper [3]. This method is essentially an extension of traditional 1-D stereo panning to two-dimensional positioning of sound using three speakers.

Using the position vectors, magnitudes, and timing information from the generated 3D impulse alongside speaker positioning data for the playback system, an array of individual impulse responses– one for each speaker– is created. The general process for each 'spike' of the 3d impulse is as follows:

1. Assuming all speakers and virtual sources are on the surface of the unit sphere, find the nearest 3 speakers to the virtual source.
2. Verify that the virtual source falls within the 'active triangle' formed by the 3 selected speakers, as illustrated in Figure 5a-b.

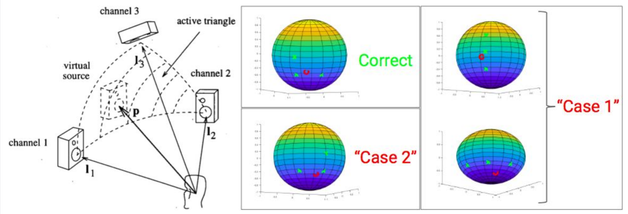

Figure 5a-b. Illustration of a virtual source within the active triangle of three loudspeakers (a, left) [3]. Visual representations of speaker selection errors due to suboptimal speaker placement in the playback room (b, right).

3. Calculate unit vectors for the virtual source and each of the three selected speakers, and perform matrix operations as detailed in Pulkki's paper [3] to calculate gain scaling factors for each speaker.
4. Update the individual impulse responses for each of the three selected speakers using magnitude and timing information from the 3-D impulse and scaling information from the VBAP calculation.

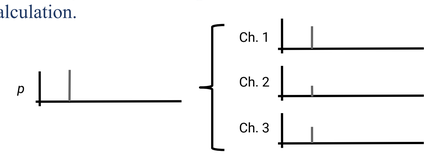

Figure 6. Visualization of panning an impulse in a situation similar to figure 5a. The relative distances from virtual source 'p' to each channel are reflected in the relative magnitudes of the channel outputs.

Once the entire 3-D impulse has been processed and each speaker in the playback system has a corresponding impulse response, the final stages of convolution and output can be performed.

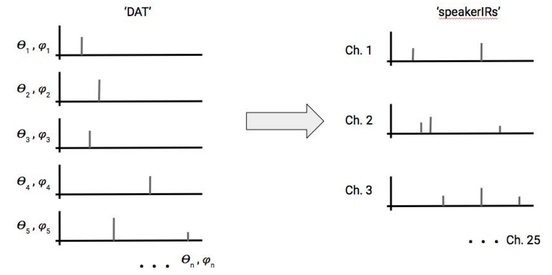

Figure 7. A visualization of the role of VBAP processing. The raw data from the 3-D impulse (left) contains an enormous number of unique angles. The output (right) reproduces all of these angles by panning across the 25 available speakers.

Convolution

Convolution is defined in the time domain as:

$$y(t) = x(t) * h(t) = \sum_{k=-\infty}^{\infty} x(k)h(t-k)$$

Equation 1

In the case of our project, *y(t)* is the reverberated output signal, *x(t)* is the mono input signal, and *h(t)* is the synthesized impulse response. This convolution operation is performed in MATLAB using the built in function, conv(). Although this function is able to successfully convolve our data, other methods of convolution are computationally less expensive and still produce the desired result.

It is well established that convolution is computed more quickly in the frequency domain

$$Y(j\omega) = X(j\omega) * H(j\omega)$$

Equation 2

For this reason, we are utilizing the MATLAB function fftfilt(). This implements *y(t) = ifft(fft(H,nfft) * fft(X,nfft)*, where nfft is the length of the fft. Using the overlap-add technique (OLA,) this function applies the impulse response to the audio input as a FIR filter. This method allows us to apply reverb to longer audio files and generate larger rooms without sacrificing computational efficiency.

Output

In MATLAB, one typically uses the function *soundsc()* for mono or stereo channel playback. We instead utilized the audioDeviceWriter object in MATLAB, which writes audio directly to the computer's sound card. This feature is highly customizable, allowing one to specify parameters such as device, sample rate, bit depth, and channel mapping. The audioDeviceWriter generates our 25-channel track, as shown in image 3.

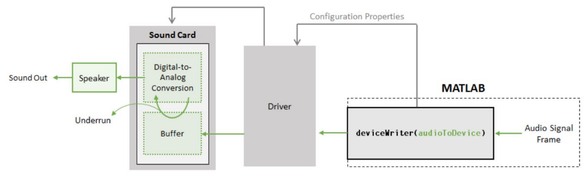

Image 2. Schematic demonstrating the flow of audio output using the deviceWriter object. [4]

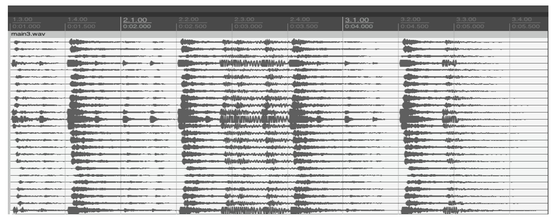

Image 3. Our output, a 25 channel track displayed in Reaper.

Citations

[1] Bocko, M (2018) Imp_Resp_w_Angle_3.m source code (Version 1.0) [Source code].
[2] B. Allen, J. (1976). Image method for efficiently simulating small-room acoustics. Acoustical Society of America Journal. 60. 9-. 10.1121/1.2003643
[3] Pulkki, V. (1997). Virtual Sound Source Positioning Using Vector Base Amplitude Panning. J. Audio Eng. Soc., Vol. 45, No. 6, 1997 June
[4] MathWorks. *Audio Device Writer Data Flow*. 2016.