SIMULATION & TESTING OF A MULTICHANNEL SYTEM FOR 3D SOUND LOCALIZATION

A thesis presented to the faculty of the Graduate School of Western Carolina University in partial fulfillment of the requirements for the degree of Master of Science in Technology.

By

Edward Albert Matthews

Director: Robert Adams, Ph.D.
Associate Professor
Department of Engineering and Technology

Committee Members:
Yanjun Yan, Ph.D., Department of Engineering and Technology
Peter Tay, Ph.D., Department of Engineering and Technology
Daniel Gonko, Commercial and Electronic Music

September 2015

TABLE OF CONTENTS

LIST OF TABLES iii
LIST OF FIGURES iv
ABSTRACT v
CHAPTER 1: INTRODUCTION 1
CHAPTER 2: LITERATURE REVIEW 4
CHAPTER 3: DESIGN & METHODOLOGY 11
3.1 Equidistant Localization 11
3.1.1 Two Channel System 11
3.1.1.1 Algorithm 13
3.1.1.2 Test Procedure 15
3.1.2 Multi-Channel Systems 18
3.1.2.1 Mathematical Derivation 18
3.1.2.2 Algorithm 19
3.1.2.3 Test Procedure 22
3.2 Non-Equidistant Localization 25
3.2.1 Mathematical Derivation 26
3.2.2 Displaying the Model 27
CHAPTER 4: RESULTS & DISCUSSION 32
4.1 Two Channel Figures and Summary 32
4.2 Multi-Channel Figures and Summary 36
4.3 Non Equidistant Results 39
CHAPTER 5: CONCLUSION & FUTURE WORK 47
REFERENCES 49
APPENDIX A: Experimental Data 51
APPENDIX B: MATLAB Code 57

LIST OF TABLES

Table A1: Two channel experimental data – participants 1 to 5 51
Table A2: Two channel experimental data – participants 6 to 10 52
Table A3: Two channel experimental data – participants 11 to 15 52
Table A4: Multi-channel Perimeter Test positions 53
Table A5: Perimeter Test data – participants 1 to 5 53
Table A6: Perimeter Test data – participants 6 to 10 54
Table A7: Perimeter Test data – participants 11 to 15 54
Table A8: Multi-channel Random Angle Test positions 55
Table A9: Random Angle Test data – participants 1 to 5 55
Table A10: Random Angle Test data – participants 5 to 10 56
Table A11: Random Angle Test data – participants 11 to 15 56

LIST OF FIGURES

Figure 1: Coordinate system for sound direction 1
Figure 2: Paths from a source (A,B) to the left and right ears 2
Figure 3: Two-channel stereophonic configuration 5
Figure 4: Binaural audio system with loudspeakers 7
Figure 5: Three speaker multi-channel configuration ($l_1=l_2=l_3$) 9
Figure 6: Stereophonic configuration formulated with vectors 12
Figure 7: Two-channel simulation example 15
Figure 8a: Two-channel experimental setup – table & chin rest 16
Figure 8b: Two-channel experimental setup – listener perspective 16
Figure 9: Three-channel simulation example 21
Figure 10: Three-channel experimental setup 22
Figure 11: Virtual source locations 23
Figure 12: Perimeter Test positions 24
Figure 13: Random Test positions 25
Figure 14a: Sound intensity at time=5.3903ms 29
Figure 14b: Sound intensity at time=5.8395ms 29
Figure 14c: Sound intensity at time=26.5024ms 30
Figure 14d: Sound intensity at time=50.3097ms 30
Figure 15: Random Angle Test results – Angle vs. Level Difference 32
Figure 16: Random Angle Test results – Perceived vs. Calculated 33
Figure 17: Random Angle Tests – Average of Responses 34
Figure 18: Random Angle Test - RMS error 35
Figure 19a: Perimeter Test results – percent of correct responses 36
Figure 19b: Perimeter Test results – percent of responses within $15°$ 37
Figure 20a: Random Test results – percent of correct responses 38
Figure 20b: Random Test results – percent of responses within 15° 38
Figure 21a: Interaural Correction Algorithm example 1 40
Figure 21b: Interaural Correction Algorithm example 2 41
Figure 21c: Interaural Correction Algorithm example 3 42
Figure 21d: Interaural Correction Algorithm example 4 43
Figure 21e: Interaural Correction Algorithm example 5 44
Figure 21f: Interaural Correction Algorithm example 6 45
Figure 21g: Interaural Correction Algorithm example 7 46

ABSTRACT

SIMULATION & TESTING OF A MULTICHANNEL SYTEM FOR 3D SOUND LOCALIZATION

Edward Albert Matthews, M.S.T.

Western Carolina University (September 2015)

Director: Dr. Robert Adams

Three-dimensional (3D) audio involves the ability to localize sound anywhere in a three-dimensional space. 3D audio can be used to provide the listener with the perception of moving sounds and can provide a realistic listening experience for applications such as gaming, video conferencing, movies, and concerts. The purpose of this research is to simulate and test 3D audio by incorporating auditory localization techniques in a multi-channel speaker system. The objective is to develop an algorithm that can place an audio event in a desired location by calculating and controlling the gain factors of each speaker. A MATLAB simulation displays the location of the speakers and perceived sound, which is verified through experimentation. The scenario in which the listener is not equidistant from each of the speakers is also investigated and simulated. This research is envisioned to lead to a better understanding of human localization of sound, and will contribute to a more realistic listening experience.

CHAPTER 1: INTRODUCTION

The human ears hear the world in 3D. In other words, we can naturally distinguish which direction a sound is coming from and how far away the source is. This process is called auditory localization. The three coordinates that must be determined when detecting the location of a sound are the azimuth, elevation, and distance. The azimuth is the angle between the source and the sagittal, or median, plane (left or right position); elevation is the angle between the source and the horizontal plane through the ears (up and down position); and distance is how far away the source is from the listener. Figure 1 shows the coordinate system for sound localization relative to a listener.

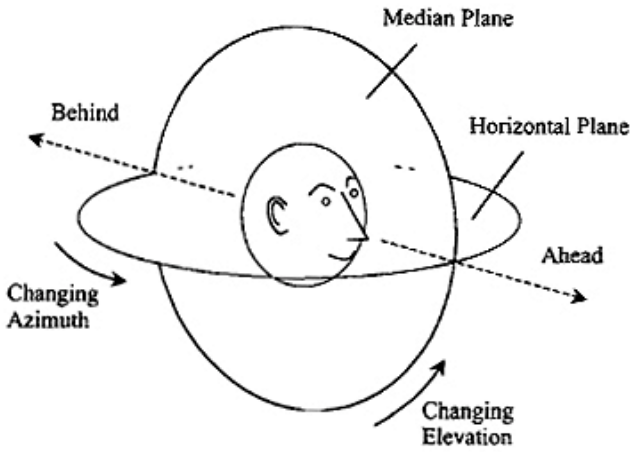

Figure 1: Coordinate system for sound direction

To determine azimuth, our ears use interaural cues, or slight differences in time and pressure of a sound as it reaches the left and right ear [1]. Interaural time difference (ITD) is the delay of the sound between the left and right ear, and interaural level difference (ILD) is the difference in sound pressure [2]. When a source is directly

in front of a person, the timing and pressure will be the same, but if the source is moved to the left or right, the sound will reach one ear slightly before the other with a higher pressure. Figure 2 shows that the paths to the left and right ear are equidistant from a source directly in front of a listener (A), but are different when the source is moved to the side (B):

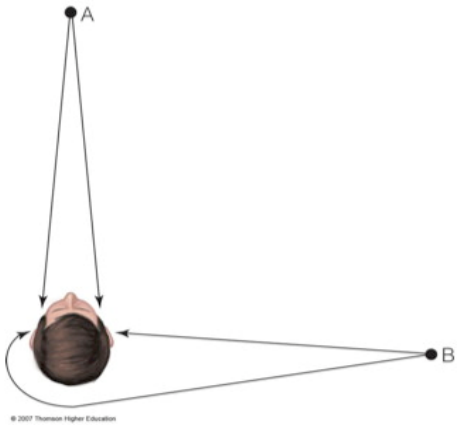

Figure 2: Paths from a source (A,B) to the left and right ears

This difference gives us the ability to localize a source to our left or right. However, if the source is anywhere in the median plane, then the distance of the paths to the ears is equal, hence there will be no interaural differences to distinguish if the sound is coming from in front of, behind, above, or below the listener. Therefore, to determine elevation we must rely on the head related transfer function (HRTF). As a sound travels from the source to the eardrum, the signal is filtered and attenuated by objects such as the head, torso, and outer ear; the differences in the intensities of frequencies from those at the source and at the eardrum make up the HRTF [3]. Neither the HRTF nor interaural cues give information on distance, so the listener must rely on the loudness of the sound compared to familiar sources to tell how far away it is. Things such as echoes

and the timbre of the sound can also aid in judging distance [3]. The combination of these cues gives us the ability to put a spatial location on a sound source without seeing it. Therefore, if we can control these parameters then we will be able to trick the mind into thinking a sound is coming from a desired location.

CHAPTER 2: LITERATURE REVIEW

There have been many advances in graphics and video that make us feel like we are a part of the movie or game that we are viewing. However, most modern stereo systems, even surround sound, cannot give us the feeling of being “in” the environment of what we are listening to. Surround sound systems, such as Dolby 5.1, generate sounds from different directions, but the source of the sound is localized at the speaker it is coming from. They are not able to make the sound appear to be coming from above or below the horizon of the speakers, or move the sound closer or farther away from the listener. 3D audio would give you the ability to place the sound anywhere in the room. This would give the listener a more realistic experience and would have many applications such as gaming, video conferencing, movies, and concerts. To accomplish this, we need to control the cues that tell the listener where the sound is coming from.

There are several techniques that have been explored to render 3D audio. The Haas effect, or precedence effect, is a psychoacoustic phenomenon that states that if two wave fronts hit the ear within a certain amount of time of each other, then the sound is perceived as a single sound (auditory event), coming primarily from the direction of the first arriving wave front [4]. The second sound influences the spatial location, but is dominated by the first even if it is slightly louder. Studies have shown that the limit of the delay before the sound breaks apart into two separate events is 5-10ms for clicks [5] and more than 50ms for speech [6]. Any sound that is delayed above these thresholds will be perceived as an echo. This phenomenon explains why we can localize the

source of a sound in a room despite the numerous reflections of the signal off its surfaces. We can take advantage of this effect to provide a solution to the lack of dimension in audio systems.

One of the simplest and most common spatial audio systems is the two channel stereophonic configuration. Figure 3 shows Pulkki's illustration of a typical stereo configuration used in [7] where $\varphi_0 = 30°$.

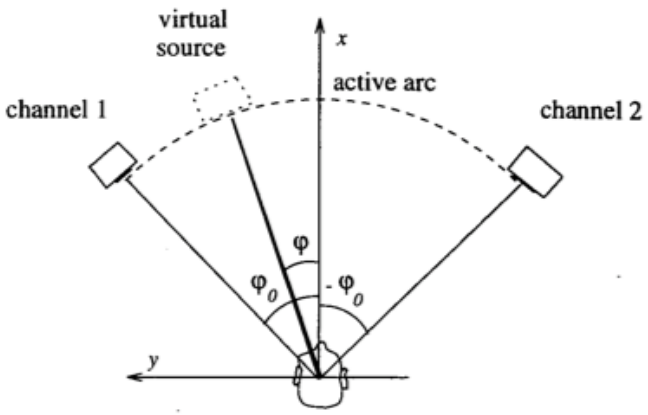

Figure 3: Two-channel stereophonic configuration

When the same signal is emitted from both channels, the sound will be perceived as a single auditory event coming from somewhere between the two speakers [7]. This event is known as a virtual, or phantom, sound source. It is possible to move the phantom source to any point on a path between the two speakers, called the active arc, simply by controlling the gain and/or delay of the speakers [8]. The radius of the active arc is defined by the distance to the speakers. Adjusting the amplitude of the sound coming from each channel is known as intensity panning, and adjusting the delay of the sound at each channel is known as time panning [2]. If there are two loudspeakers positioned symmetrically to the median plane, their gains are equal, and there is no delay, then the virtual source will be perceived to be directly in the center. The source

can move along the arc between the two speakers by controlling the ratio of their gains. As one speaker is made louder than the other, the virtual source will move closer to that speaker.

Many authors have investigated these panning techniques (eg. [9], [10], [11]). According to the Haas effect and summing localization [2], as long as the delay does not surpass the echo threshold, then the sound will be heard as one event based on the direction of the first arriving wave. However, there are some drawbacks to these panning techniques. The phantom source can be positioned anywhere between the two speakers, but lacks the ability to move anywhere outside of this arc [12]. The position of the listener is also very restricted. The listener must be positioned so that they are at an equal distance from each of the speakers, and are oriented so that the speakers are symmetrical on the median plane. This position is known as the “sweet spot” [12]. If the listener moves outside of the sweet spot, or rotates their body or head, then the perceived sound location will be incorrect [13], [14]. Because of these flaws, many techniques have been investigated to move the source outside of this boundary [15], increase the size of the sweet spot [8], or make the sweet spot move with the listener [16], [17].

Bauer [10] introduced a method to move the sound outside of the stereo boundary: crosstalk cancellation. Gardner [17] describes crosstalk cancellation as “inverting the transmission paths that exist from the speakers to the listener” to cancel “crosstalk” from the right speaker to the left ear and vice versa. Systems that implement a binaural synthesizer with a crosstalk canceller are known as binaural audio systems. The goal of these systems is “to reconstruct the acoustic pressures at the listener’s ears

that would result from the natural listening situation to be simulated" [17]. Song [16] discusses the two major blocks of binaural audio systems: the binaural synthesizer B, which computes the sound that should be heard by the listener's ear, and the crosstalk canceller H, which compensates for the transmission path. The block diagram they provide is shown below:

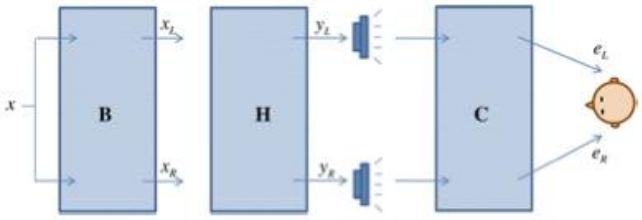

Figure 4: Binaural audio system with loudspeakers

The binaural synthesizer uses the monaural input signal and HRTFs to generate the output signals x_L and x_R. The crosstalk canceller H is an inverse filter of C, which represents the acoustic paths between the speakers and the ears. Several studies have been carried out to improve the methods of filtering in crosstalk cancellation, and many successful approaches have been verified (eg. [15], [18], [19], [20], [21]). However, since B, H, and C (represented by matrices) are each calculated based on the listener being in the sweet spot, the 3D effect is degraded when the listener moves. Several papers (eg. [16], [17]) have presented "dynamic binaural systems" to fix this problem. These systems incorporate head tracking to find the position of the listener's head, and update the binaural synthesis and crosstalk matrices according to its new position. These systems have shown that it is possible to make a 3D audio system that compensates for movement.

Another approach to generating 3D audio is by reproducing the waveforms of an actual sound image through headphones. This is usually done using special recording techniques where microphones are placed in a listener's, or a dummy heads, ears to measure HRTFs, which can then be used to recreate desired signals. Wightman [22] performed experiments on eight subjects to compare localization of sound presented in free field to headphones. The subjects were first asked to identify the apparent positions of sounds delivered from 36 different positions through six loudspeakers. For each position, stimuli for the headphone testing had to be produced by digitally filtering signals using the subjects corresponding HRTF and ear canal transfer function (ECTF). The stimuli were then presented to the listener through headphones and they were asked to identify the perceived location. According to his results, the spatial positions identified when the sound was played through the headphones matched positions identified in free field. There have been several other studies that have validated that headphone delivered stimulus can reproduce a free field source (eg. [12], [23], [24], [25]). In addition, most of these studies have pointed out that headphone systems cannot fully recreate the entire free field image, especially in the front, back, and elevated positions, and have proposed methods to increase the spatial extent of the perceived sound image. Although these studies have improved the performance of 3D headphone systems, there is still the obvious drawback of being limited to the one listener.

The techniques that have been presented thus far have all shared a common disadvantage of being limited to one listener, or being confined to a small listening area. In theory, to achieve 3D audio that can be enjoyed by a larger audience, a multi-channel

system would need to be implemented. By adding more speakers, there are more paths on which the virtual source can move [7]. Many of the same techniques that were used to simulate spatial sound with two channels can be applied to these systems. It was shown in [26] that crosstalk cancellation and equalization (CTCE) could be applied to systems using more than two loudspeakers. Furthermore, it was justified by simulations that by using more loudspeakers "a more robust CTCE system that is less sensitive to errors in the measured impulse responses" [26] is achievable. Other systems have also been investigated where additional speaker[s] are included above the horizontal plane, such as Auro 11.1 [27] and the 22.2 format discussed in [28]. Pulkki presents a simpler model in [7] where a single elevated speaker is introduced to the stereo configuration from Figure 3, as shown in Figure 5.

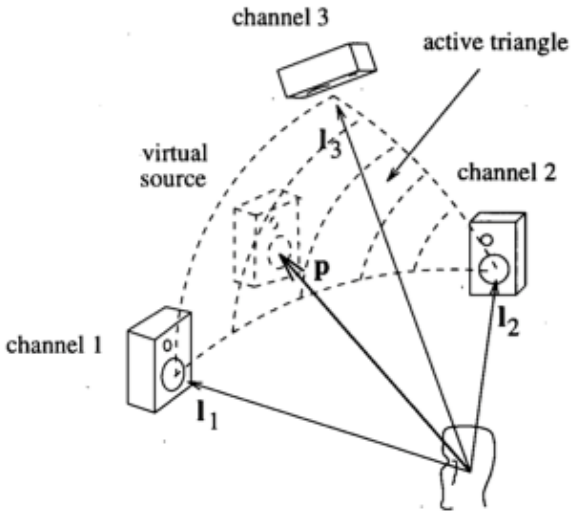

Figure 5: Three speaker multi-channel configuration (l_1=l_2=l_3)

The elevated speaker is the same distance from the listener as the other loudspeakers. As you can see, the three speakers form a section of a 3-D sphere known as the "active triangle" on which the virtual source can be positioned [7]. Pulkki went on to explain

how this system can be expanded with more loudspeakers, where the speakers form bases in groups of three. He found that the “maximum error in the virtual source localization is proportional to the dimensions of the active region. Therefore when good localization accuracies on a large listening area are desired, the dimensions of the active regions must be decreased. This is done by applying more loudspeakers on the desired region of the sound field”. Based on his work it is easy to say that the more speakers there are in a system, the better. However in practical application, as the number of speakers increases, so does the cost and amount of space required for the system. Therefore, for the proposed research the number of channels will be limited to three.

CHAPTER 3: DESIGN & METHODOLOGY

In this research effort two and three channel localization systems were simulated then tested in a lab environment, and a non-equidistant interaural correction system was simulated. This chapter presents the design and methodology of the simulations and experiments.

3.1 Equidistant Localization

In this section our investigation of the vector based amplitude panning (VBAP) discussed in Pulkki's paper [7] will be presented. Amplitude panning is an audio technique where the same signal is played over two or more speakers that are equidistant from an observer. Since the signals are the same and there is no interaural time delay, the observer will perceive the illusion of a single virtual source. The position of the virtual source depends on the locations of the speakers, and the relation between the amplitudes of the signals they produce. The amplitude of the signals can be controlled by adjusting the gains of each speaker. The following two sections will discuss the mathematical derivation, algorithm development, and testing procedure for two and three channel systems, which both implement amplitude panning.

3.1.1 Two Channel System

Figure 3 can be reformulated as a two-dimensional vector base as shown in Figure 6

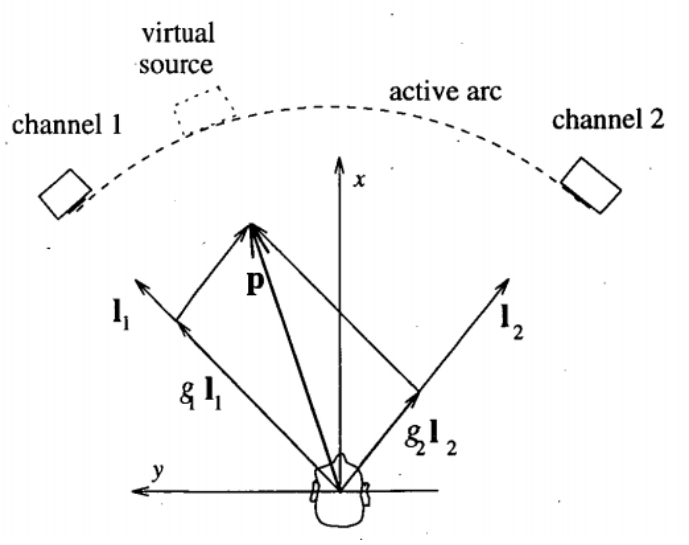

Figure 6: Stereophonic configuration formulated with vectors

where $l_1$and l_2 are the position vectors for the left and right speakers, and p is the virtual source vector. The speaker vectors can be expressed using Equation 1

$$l_m = \left[l_{mx}\ l_{my}\right]^T \qquad (1)$$

where m represents the channel number (1 or 2), l_{mx} is the x component of the vector, and l_{my} is the y component of the vector. Therefore the equations for the left and right speaker vectors are given by

$$l_1 = \left[l_{1x}\ l_{1y}\right]^T \qquad (2)$$

$$l_2 = \left[l_{2x}\ l_{2y}\right]^T \qquad (3)$$

The distance to each speaker is equal, therefore the vectors have the same length. The gain factors of the left and right speakers, g_1 and g_2, are nonnegative scalar variables in the range of zero to one. To keep the loudness of the virtual source constant the gain factors must be normalized using Equation 4

$$g_1^2 + g_2^2 = C \qquad (4)$$

where C is the volume of the virtual source. As C increases, the virtual source is perceived to move closer to the observer.

The vector pointing towards the phantom source can be represented as a linear combination of the speaker vectors and their respective gains

$$p = [p_1 \ p_2]^T = g_1 l_1 + g_2 l_2 \quad (5)$$

The equation may also be written in matrix form

$$p^T = g L_{12} \quad (6)$$

where $g = [g_1 \ g_2]$ and $L_{12} = [l_1 \ l_2]^T$. This equation calculates the x and y components of the phantom source vector. Therefore, if we are given the location and gains of each speaker, we can calculate the position of the virtual source.

3.1.1.1 Algorithm

Rather than finding the location of the virtual source based on speaker parameters, the algorithm calculates the proper gain factors needed to put the virtual source in a desired location. The algorithm calculates the proper gain factors by taking the inverse of Equation 6, which yields

$$g = p^T L_{12}^{-1} = [p_x \ p_y] \begin{bmatrix} l_{1x} & l_{1y} \\ l_{2x} & l_{2y} \end{bmatrix}^{-1} \quad (7)$$

Thus the equation calculates the gain factors given the vectors pointing to the left and right speakers, and the phantom source. The user is asked to input the location of the speakers by defining the distance to the speakers and the angle from the x axis to the left and right speakers. The user is also asked to input the angle from the x axis to the desired virtual source position**.** Using these values, the algorithm calculates the vectors pointing toward the speakers and desired virtual source using the following equations

$$l_{mx} = s\cos\theta_m \quad (8)$$

$$l_{my} = s\sin\theta_m \quad (9)$$

$$p_x = s\cos\theta_p \quad (10)$$

$$p_y = s\sin\theta_p \quad (11)$$

where m is the channel number (1 or 2), s is the distance to the speakers, and θ_m and θ_p are the angles from the x axis for each speaker and the virtual source, respectively. The algorithm checks user input to make sure that none of them are out of bounds (e.g. if the virtual source lies outside of the active arc)**.** Using Equation 7, the algorithm calculates the gain factors of the left and right speakers, and plots the speakers and virtual source. Figure 7 is an example of a plot generated that shows the speaker locations, vectors, and calculated virtual source. For this example, the left and right speakers were placed 8 feet from the listener at $\pm 45°$ from the x axis, and the desired virtual source position was $15°$**.** The algorithm calculated the left and right gains to be $g_1 = 0.866$ and $g_2 = 0.5$, respectively.

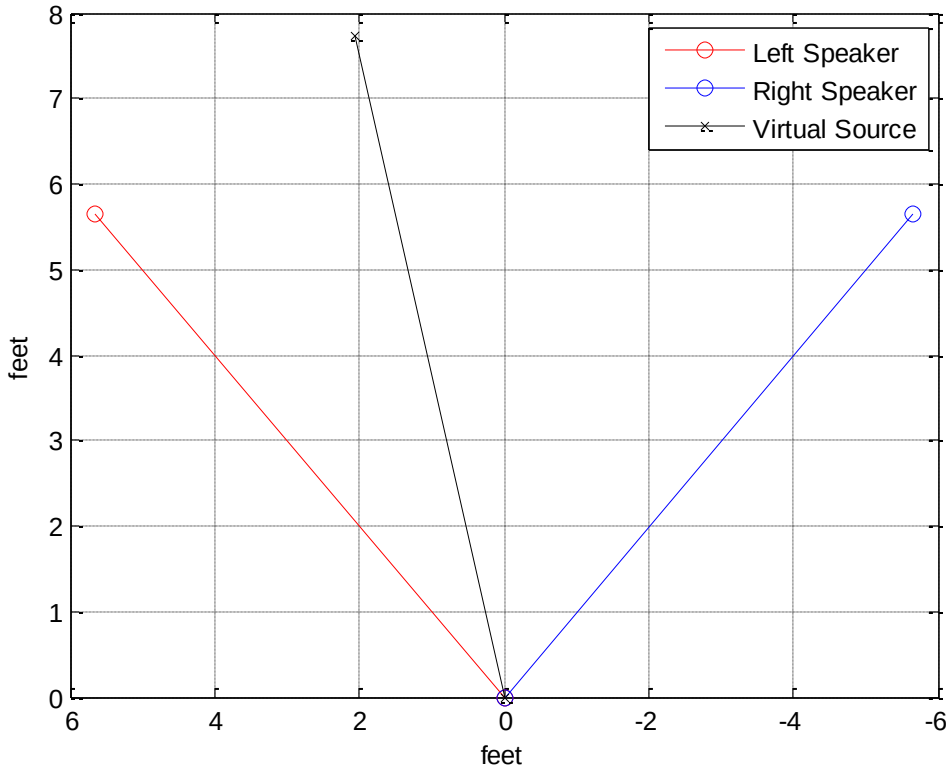

Figure 7: Two-channel simulation example

3.1.1.2 Test Procedure

To verify that the algorithm calculated the proper gain factors, experimental data was collected from 15 participants. IRB approval was obtained in order to conduct testing involving volunteers. The experiment required a quiet room with sufficient space, and a test setup to generate the acoustical signals. The experimental setup is shown in Figure 8a and Figure 8b below

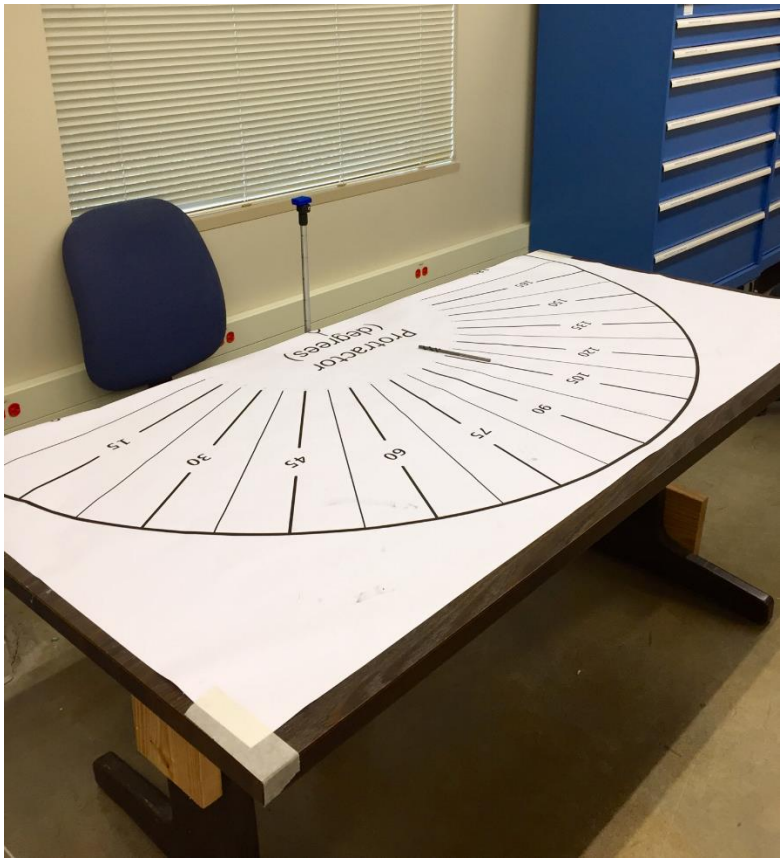

Figure 8a: Two-channel experimental setup – table & chin rest

Figure 8b: Two-channel experimental setup – listener perspective

The observer sat a table covered by a protractor to indicate angles. A chin rest was used to minimize head movement during the experiment. A base signal was produced and the corresponding gain factors calculated by the algorithm (Equation 7) were applied to each channel. The base signal used for testing was a 400 Hz sine wave with a sampling frequency of 44.1 kHz played for 2 seconds. A Sherwood RX-4109 stereo system amplified the signal which was output to two Klipsch B-3 bookshelf speakers. The left and right speakers were placed at $\pm 45°$ from the x axis at a distance of 8 feet from the observer. These values were used as inputs to the algorithm. Three variations of the algorithm were used: Training Program, Discrete Angle Test, and Random Angle Test.

First, a Training Program was run to verify the speakers were working properly, and to familiarize the participant with the signal and perception of a virtual source. The Training Program incremented the desired virtual source position between the left and right speakers in steps of $15°$. At each angle the algorithm would calculate the proper gain factors and play the resulting tone. The participant was informed of the locations that would be played before testing, and then asked if they perceived the movement of sound after the program was complete.

For the Discrete Angle Test, 20 trials were run where the desired virtual source position was selected from the angles $0°$, $\pm 15°$, $\pm 30°$, or $\pm 45°$. For each trial one of these angles was chosen as the virtual source position from a randomized list, the gain factors were calculated, and the resulting tone was played. The participant was instructed to look forward throughout the test, and encouraged to place their hand or a pointer in the direction of the perceived sound. The tone could be replayed if the

participant wanted to hear it again. For each trial the participant was asked to identify which of the angles were closest to the perceived sound, and the value was recorded.

The Random Angle Test was very similar to the Discrete Angle Test, however the virtual source position was set based on a random angle selected from twenty evenly spaced values from $-45°$ to $45°$. The participant was instructed to identify the closest angle, within a few degrees, to where they perceived the virtual source to be, and the value was recorded.

3.1.2 Multi-Channel Systems

A multi-channel system is a system that uses more than two channels. To achieve three-dimensional sound using the VBAP technique, a multichannel system must be used. An additional speaker is placed above the two dimensional plane created by the listener and the left and right speakers from the stereophonic configuration, as shown in Figure 5. The elevated speaker must be the same distance from the listener as the other two speakers. This forms a region of a sphere on which the virtual source can be moved, called the active triangle. More speakers can be used in this technique, however only up to three speakers will be active (producing sound) at one time. For this reason, only a three speaker system was investigated in our research.

3.1.2.1 Mathematical Derivation

The formulation of the three-dimensional system is exactly the same as the stereophonic configuration, however there will be three speaker vectors and gain factors, rather than two, and each vector will have an x, y, and z component. Each speaker vector can be expressed using the Equation 12

$$l_m = \left[l_{mx}\ l_{my}\ l_{mz}\right]^T \quad (12)$$

where m represents the channel number (1, 2, or 3), l_{mx} is the x component of the vector, l_{my} is the y component of the vector, and l_{mz} is the z component of the vector. The gain factors of the left, right, and top speakers, $g_1\ g_2$ and g_3, are nonnegative scalar variables in the range of zero to one. To keep the loudness of the virtual source constant, the third gain factor must be added to the normalizing equation (Eq. 4)

$$g_1^2 + g_2^2 + g_3^2 = C \quad (13)$$

where C is the volume of the virtual source. As C increases, the virtual source is perceived to move closer to the observer.

The three-dimensional vector pointing towards the phantom source can be represented as a linear combination of the speaker vectors and their respective gains

$$p = [p_1\ p_2\ p_3]^T = g_1 l_1 + g_2 l_2 + g_3 l_3 \quad (14)$$

Or, in matrix form

$$p^T = g L_{123} \quad (15)$$

where $g = [g_1\ g_2\ g_3]$ and $L_{123} = [l_1\ l_2\ l_3]^T$. This equation calculates the x, y, and z components of the phantom source vector. Therefore, if we are given the location and gains of each speaker, we can calculate the position of the virtual source.

3.1.2.2 Algorithm

The algorithm was also modified to calculate proper gain factors given the speaker parameters and desired virtual source position in three dimensional space. The input parameters were the distance to the speakers, angle from the x axis of the left, right, and desired virtual source position, and the elevation of the top speaker and

desired virtual source position**.** The x, y, and z components for each speaker vector are calculated using following equations

$$l_{mx} = s\cos\theta_m \quad (16)$$

$$l_{my} = s\sin\theta_m \quad (17)$$

$$l_{mz} = s\sin\phi_m \quad (18)$$

$$p_x = s\cos\theta_p \quad (19)$$

$$p_y = s\sin\theta_p \quad (20)$$

$$p_z = \sqrt{p_x^2 + p_y^2}\tan\phi_p \quad (21)$$

where m is the speaker number (1, 2, or 3), s is the distance to the speakers, θ_m and θ_p are the angles from the x axis for each speaker and the virtual source, respectively, and ϕ_m and ϕ_p are the elevation angles for each speaker and the virtual source, respectively. The algorithm checks the user inputs to make sure that none of them are out of bounds (e.g. if the virtual source lies outside of the active triangle)**.** Then, using inverse of Equation 15 the gain factors of the left, right, and top speaker are calculated.

$$g = p^T L_{123}^{-1} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix} \begin{bmatrix} l_{1x} & l_{1y} & l_{1z} \\ l_{2x} & l_{2y} & l_{2z} \\ l_{3x} & l_{3y} & l_{3z} \end{bmatrix}^{-1} \quad (22)$$

If the largest gain factor is greater than 1, then each gain is divided by that factor so that they are all are between zero and one. To ensure that the loudness is always the same (regardless of input parameters), a scaling factor is calculated

$$n = \sqrt{g_1^2 + g_2^2 + g_3^2} \quad (23)$$

Each gain factor is then divided by n. This places the virtual source on the surface of a sphere surrounding the listener, so it always sounds like it is the same distance away. When a desired position is input, the algorithm calculates the normalized gains, and a plot showing the speaker locations and vectors, as well as the virtual source and an approximation of the active triangle is generated. Figure 9 is an example of the three-dimensional plot, where the left and right speakers are placed 8 feet from the listener at $\pm 45°$ azimuth $0°$ elevation, the top speaker at $0°$ azimuth $42°$ elevation, and a desired virtual source position of $-15°$ azimuth $14°$ elevation. The algorithm calculated the gains for this example to be $g_1 = 0.3042, g_2 = 0.6702, \text{and } g_3 = 0.3726$

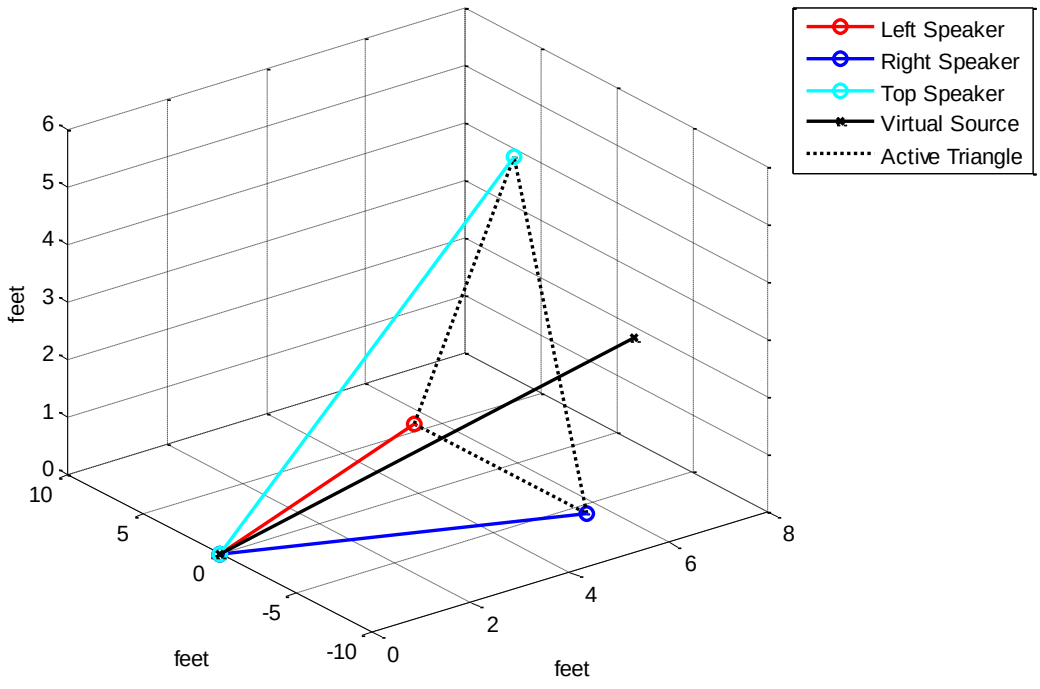

Figure 9: Three-channel simulation example

MATLAB does not have the ability to play a sound with more than two channels, therefore an audio file (.wav) for each speaker with its corresponding gain factor had to

be generated. The same base signal as in the stereophonic configuration was used, a 400 Hz sine wave with a sampling frequency of 44.1 kHz played for 2 seconds.

3.1.2.3 Test Procedure

To test the 3D algorithm, the test procedure had to be modified. The same protractor, chin rest, and left and right speakers, placed 8 feet from the listener at $\pm 45°$ azimuth $0°$ elevation, were used. A third Klipsch B-3 bookshelf speaker was added at $0°$ azimuth $42°$ elevation, also 8 feet from the listener.

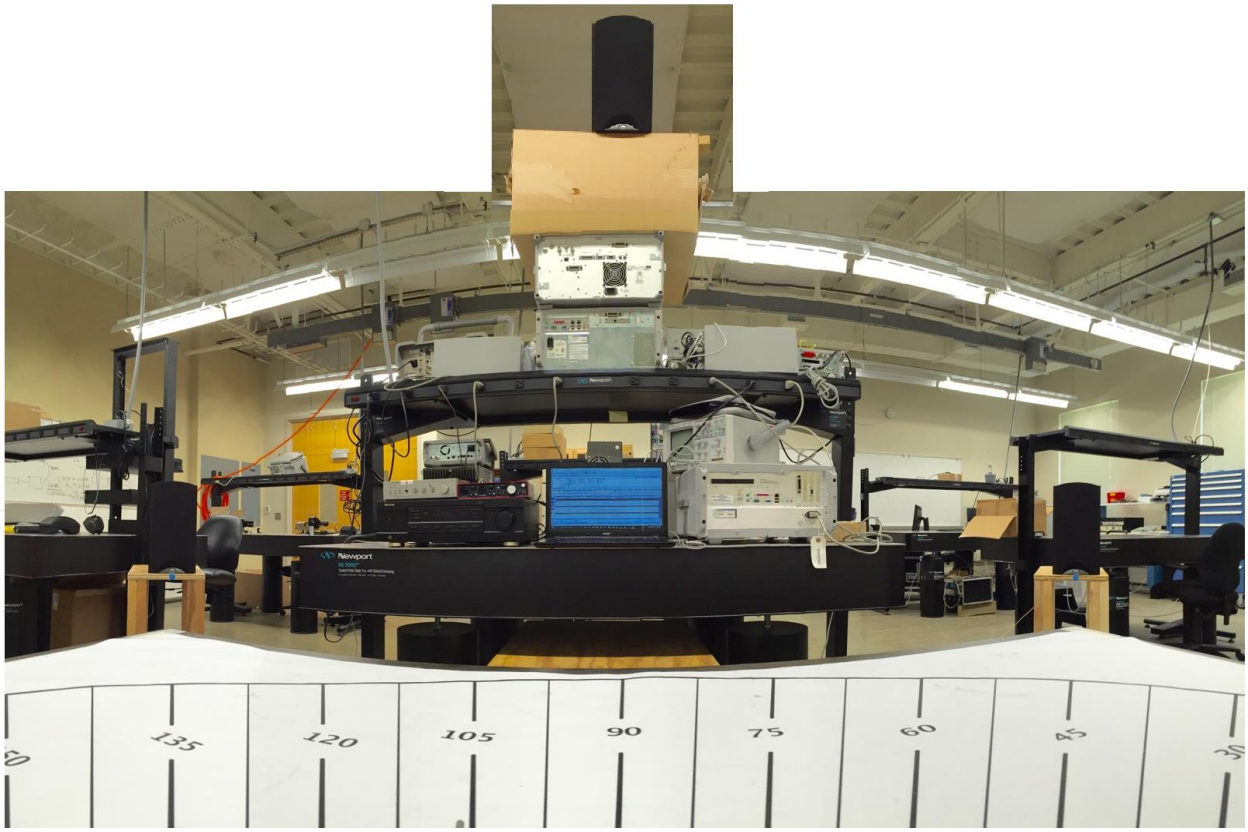

Figure 10: Three-channel experimental setup

Since the computer's sound card and outputs can only support two channels, an audio interface was needed to control all three speakers at once. An audio interface connects to the computer via USB and acts as an external sound card with extra inputs/outputs, features, and better quality. For this experiment the Focusrite Scarlett

2i4 was chosen, which has four independent outputs. The output from the interface needed to be amplified before going to the speakers. A Sherwood RX-4109 stereo system (used for the left and right speakers) and a Realistic SA-150 integrated stereo amplifier (used for the top speaker) were adjusted to calibrate the outputs to the same amplitude using a four channel oscilloscope.

A digital audio workstation (DAW), SoundForge Pro 11, was used to play the audio files made by the algorithm and communicate with the audio interface. A DAW is a computer program designed to record, mix, edit, and play audio files. Before the experiment, the algorithm was used to produce audio files that placed the virtual source in 16 different locations inside the active triangle. These files were then imported into SoundForge. The 16 locations, shown in Figure 11, had azimuth angles of $0°, \pm 15°, \pm 30°, \text{or } \pm 45°$ and elevation angles of $0°, 14°, 28°, \text{or } 42°$.

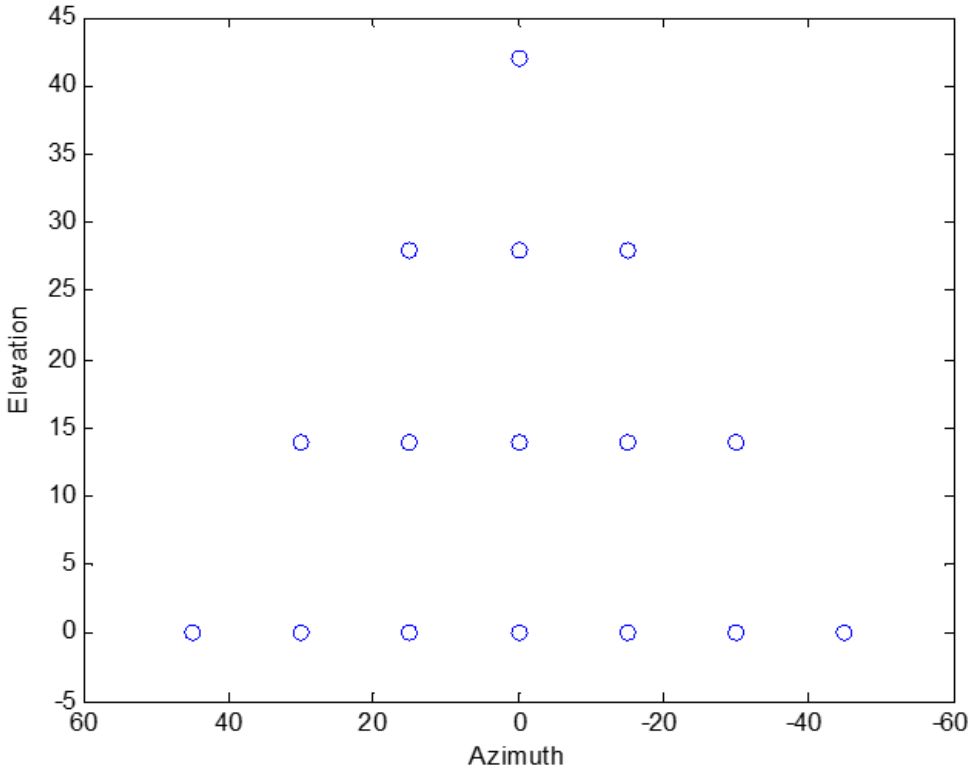

Figure 11: Virtual source locations

For multi-channel testing there were two experiments: the Perimeter Test and the Random Test. Before testing, each participant went through a training program. A number was assigned to each virtual source position, and the sound for that point was played for the listener. For the Perimeter Test, only points on the perimeter of the triangle were used. These locations are shown in Figure 12.

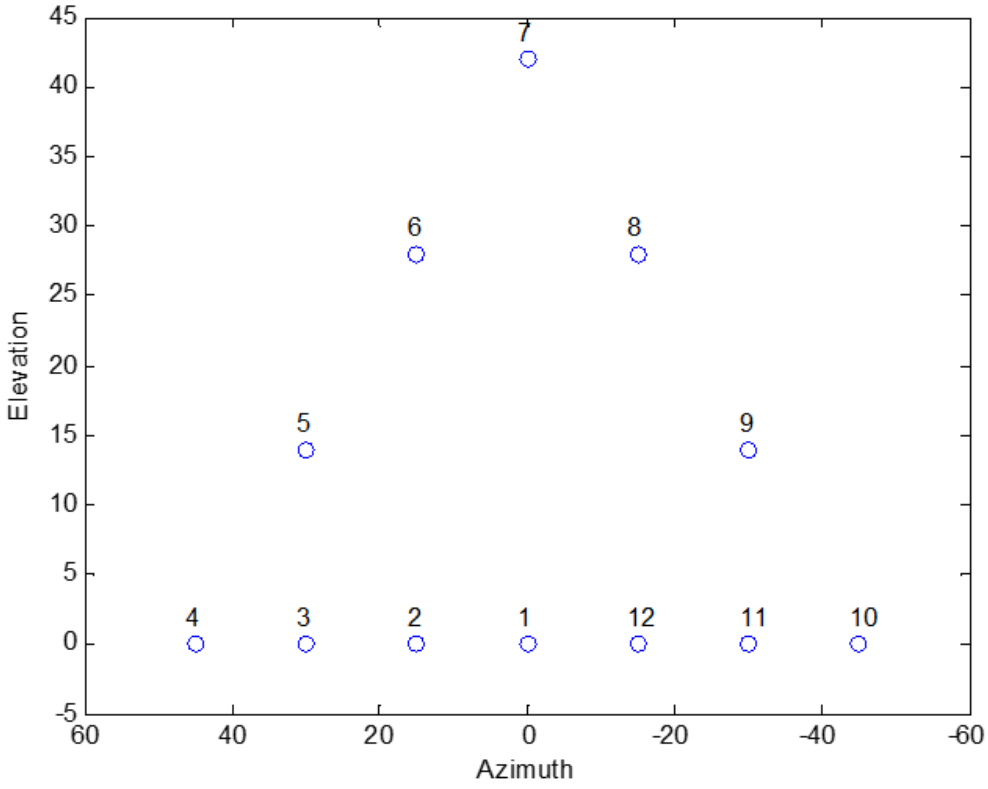

Figure 12: Perimeter Test positions

The sounds were played from one to twelve for training, and then the order was randomized for testing. For each location, the sound was played twice and the listener was asked to identify which point was closest to where they perceived the sound to be coming from.

The Random Test was conducted the same as the Perimeter Test, however all 16 points were used. The order for the Random Test training is shown in Figure 13

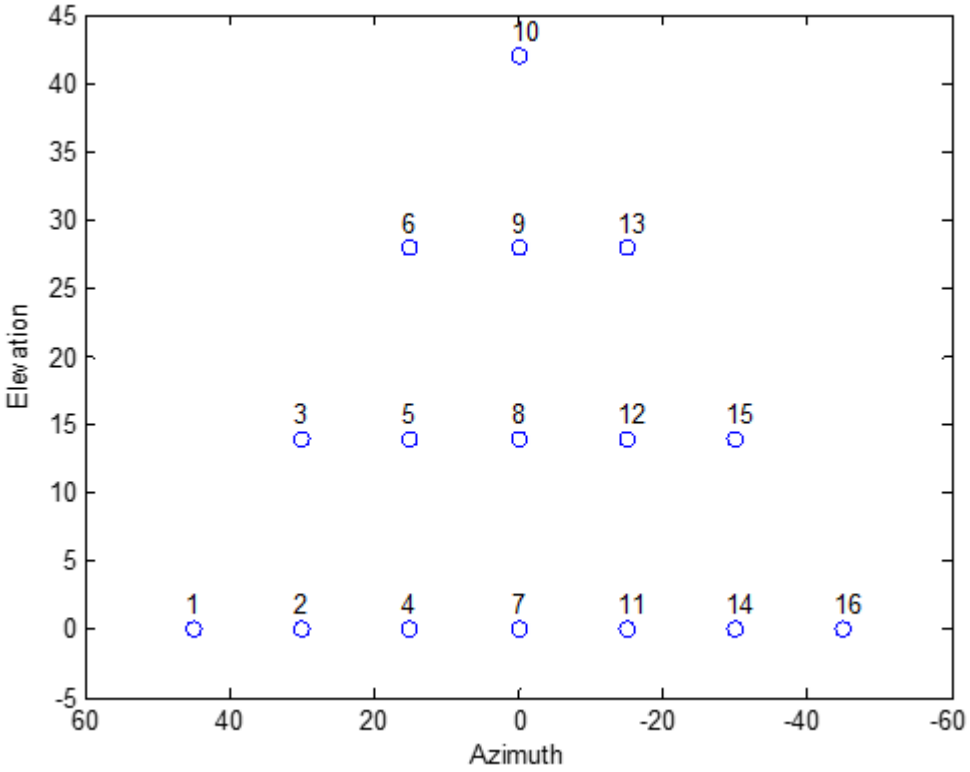

Figure 13: Random Test positions

The sounds were played from one to sixteen for training, and then the order was randomized for testing. For each location, the sound was played twice and the listener was asked to identify which point was closest to where they perceived the sound to be coming from.

3.2 Non-Equidistant Localization

Up to this point the listener is required to be in a certain spot for the algorithm to work. This region, which is equidistant from all the speakers, is called the sweet spot. However in many applications an observer may not be the same distance from all

speakers, or would possibly be moving to different locations. If the listener is not in the sweet spot, interaural time and level differences will be introduced. To compensate for the differences an interaural correction algorithm was made.

3.2.1 Mathematical Derivation

The interaural correction algorithm receives coordinates for the speakers and the listener, and angle from the x axis of the desired phantom source as inputs. Using these values, it adjusts the gain and delay of the speakers so that the sound waves reach the listener at the same time and with the same intensity. To do this the distance from the observer to the left and right speakers must be calculated and compared using the distance formula

$$d_{L,R} = \sqrt{\left(x_{L,R} - x_o\right)^2 + \left(y_{L,R} - y_o\right)^2} \qquad (24)$$

where $d_{L,R}$ are the distances in feet to the left and right speakers, $(x_{L,R}, y_{L,R})$ are the x and y coordinates of the left and right speakers, and (x_o, y_o) are the x and y coordinates of the observer. Whichever speaker is closer to the observer must be delayed, and the other speaker's gain must be multiplied by a factor to increase its volume. Sound radiates from each speaker in the form of a sphere. The intensity of sound at a given distance from a speaker is the original intensity divided by the area of the sphere. Since the area of a sphere is proportional to the square of its radius, the intensity of sound is inversely proportional to the squared distance from the speaker. This is known as the inverse square law

$$\text{Intensity} \propto \frac{1}{\text{distance}^2} \qquad (25)$$

Using this law, the factor needed to compensate for the interaural level distance can be calculated as follows

$$I_L = \frac{k}{d_L^2} \quad (26)$$

$$I_R = \frac{k}{d_R^2} \quad (27)$$

$$g_{ratio} = \frac{I_L}{I_R} = \left(\frac{d_R}{d_L}\right)^2 \quad (28)$$

$$G_{corrected} = \begin{bmatrix} g_L & g_R * g_{ratio} \end{bmatrix} \quad (29)$$

where $I_{L,R}$ are the left and right sound intensities, and k is a constant of proportionality.

To calculate how much the closer speaker must be delayed, the difference in distances is divided by the speed of sound

$$t_d = \frac{d_R - d_L}{340.29} \quad (30)$$

where t_d is the time delay in seconds, and 340.29 is the speed of sound in feet per second.

3.2.2 Displaying the Model

To display this model, a simulation of the sound waves moving from each source to the observer's location was created. First, equations for the lines between the speakers and the observer were found

$$y = mx + b \quad \Rightarrow \quad \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} x_{L,R} & 1 \\ x_o & 1 \end{bmatrix}^{-1} \begin{bmatrix} y_{L,R} \\ y_o \end{bmatrix} \quad (31)$$

where m is the slope of the line, and b is it's y-intercept. Each line was divided into 100 points (x,y) which were converted to pixel locations (row,column) using the following formulas

$$row = \frac{y - y_{max}}{y_{min} - y_{max}} * (N_{rows} - 1) + 1 \quad (32)$$

$$col = \frac{x - x_{max}}{x_{min} - x_{max}} * (N_{cols} - 1) + 1 \quad (33)$$

where $x_{min}, x_{max}, y_{min}, y_{max}$ are the extremes of the lines between the speakers and observer, and N_{rows}, N_{cols} are the dimensions of the display. To show the wave's progression, a video file was made where each frame was an image showing the wave's intensity and location along the line. The intensity of sound was represented by an 8-bit full color scale image, where red represented highest intensity and blue represented lowest intensity. Equation 29 was used to calculate the initial intensities of the sound waves at each source, and the pixels corresponding their positions were set to the resulting color value. As the wave moved toward the observer, pixels along the line would change color based on the intensity at that point. Since the intensity decreases with the square of distance, the values drop off very quickly. Therefore, we took the log of the intensities to linearize the color scale using the following equation.

$$\text{Color Intensity} = \log_{10}(\text{Sound Intensity} + 0.1) + 1 \quad (34)$$

Figure 14a, Figure 14b, Figure 14c, and Figure 14d show the sound intensity in color as the sound moves along the path from each speaker toward the observer. Four time frames are presented in these figures. In the example the coordinates of the left speaker are (10,20), the right speaker is at (20,20), and the listener is at (12,5). The interaural correction algorithm calculated the corrected gains to be $g_1 = 0.53555$ and $g_2 = 0.67586$, and for the time delay of the left speaker to be $t_d = 5.4872\ ms$.

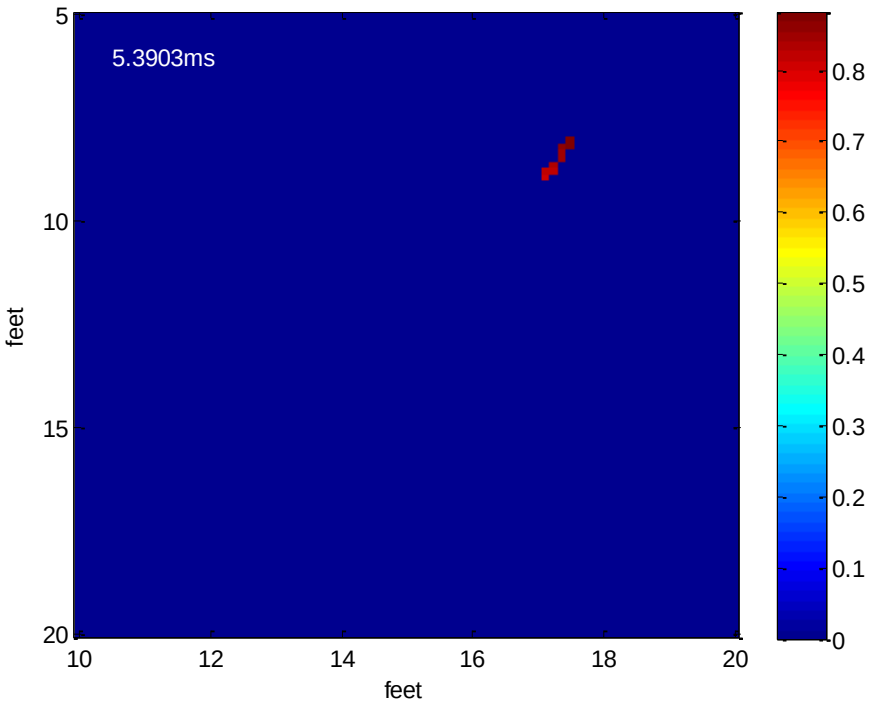

Figure 14a: Sound intensity at time=5.3903ms

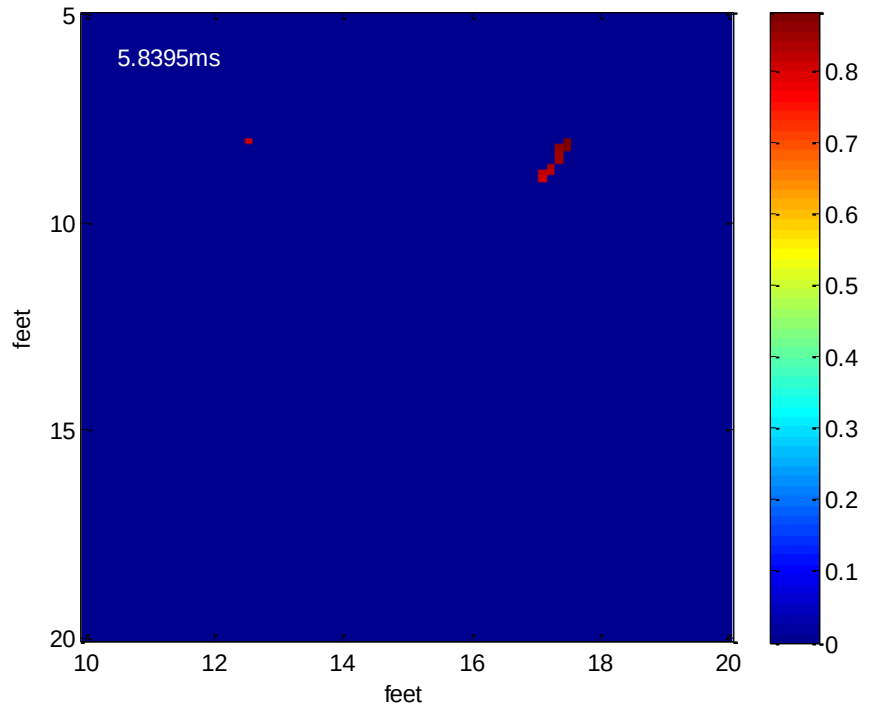

Figure 14b: Sound intensity at time=5.8395ms

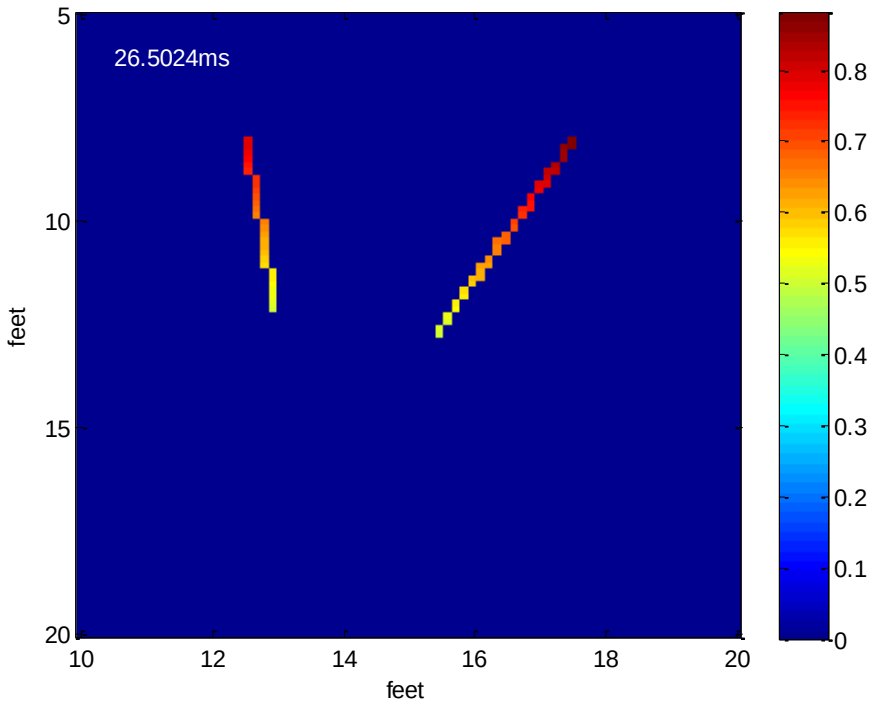

Figure 14c: Sound intensity at time=26.5024ms

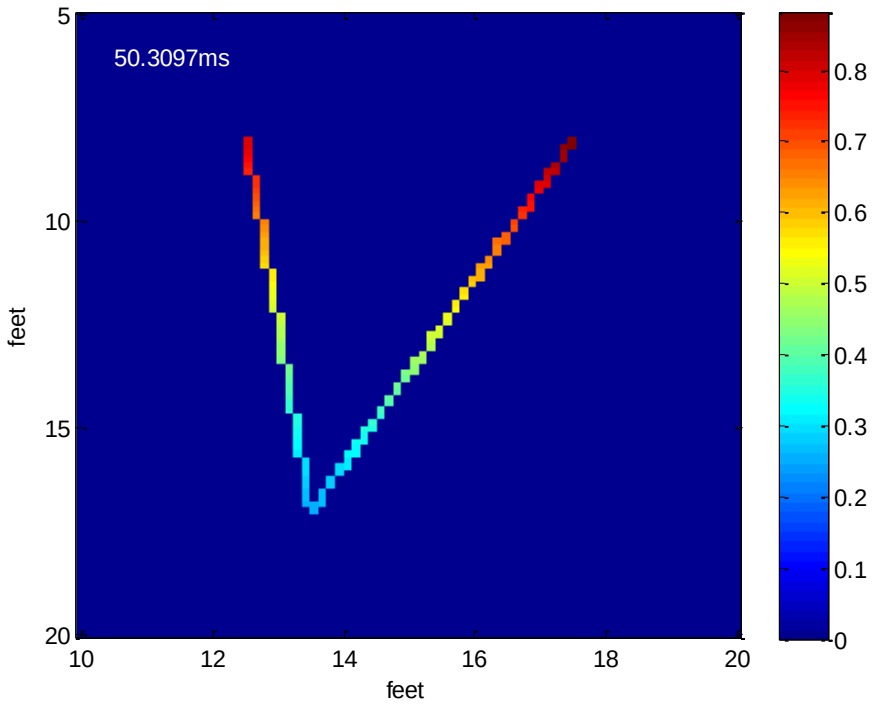

Figure 14d: Sound intensity at time=50.3097ms

The number in the upper left hand corner displays how much time has elapsed since the preliminary wave left the source. Figure 14b shows the intensities right after the delay, when the preliminary wave has reached the point that is in line with the left speaker, and the left speaker initially plays the sound. As you can see, the sound intensities are the same at this point and are equivalent as they move towards the observer, decreasing with distance from the source. If the two lines reach the listener at the same time with the same intensity (color) then the sound will be perceived directly in front of them, even though they are closer to one speaker. The colors in Figure 14d show that this is true for the example.

CHAPTER 4: RESULTS & DISCUSSION

In this chapter the results of the two and three channel localization testing, as well as non-equidistant simulation results, are presented. For the two and three channel localization tests, experimental data was collected from participants and put into tables. This data can be found in the Appendix section A

4.1 Two Channel Figures and Summary

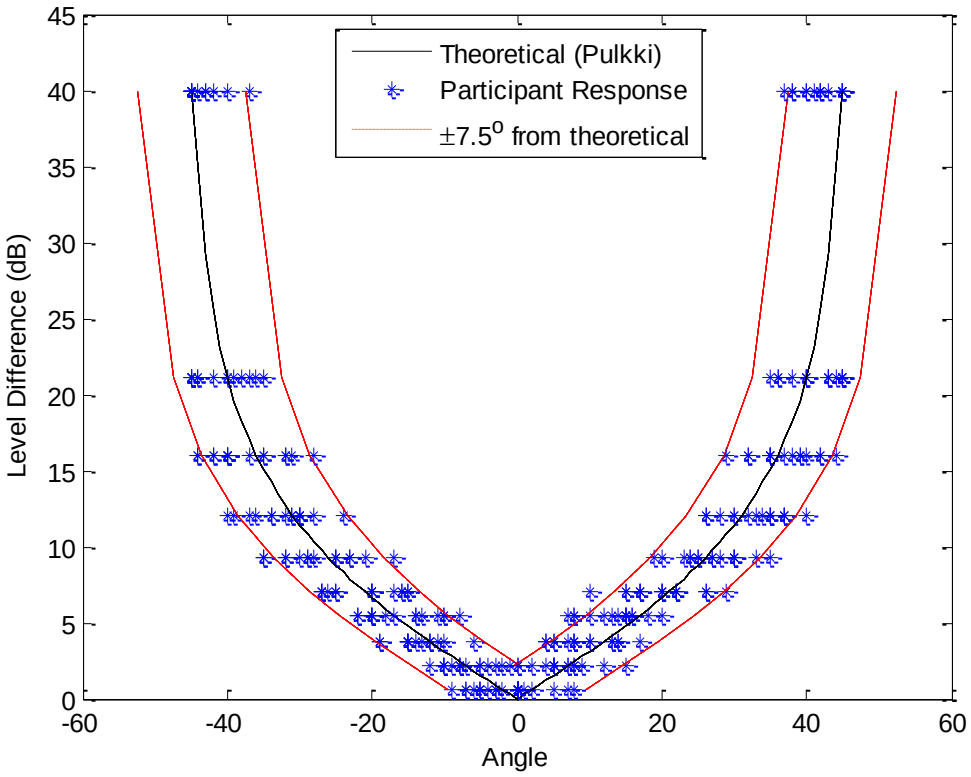

Figure 15: Random Angle Test results – Angle vs. Level Difference

Figure 15 shows how the level difference in decibels between the left and right speaker compares to the calculated and perceived angles of the virtual source. The level difference is found by converting the calculated gain values (Equation 7) to

decibels and taking the absolute value of their difference. The solid black curves are the theoretical angles corresponding to each level difference as derived in [7], the dotted red curves are ±7.5° from the calculated values, and the blue asterisks are the perceived values heard by listeners in the experiment. As you can see, the majority of participants perceived the sound to be coming from within 7.5° of the theoretical value.

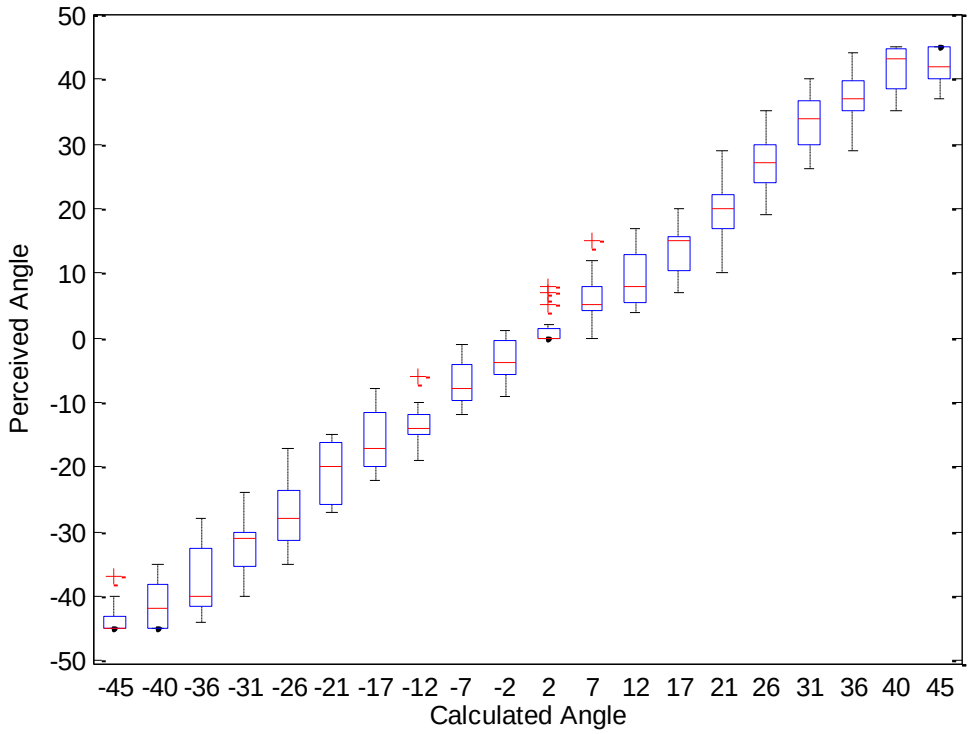

Figure 16: Random Angle Test results – Perceived vs. Calculated

Figure 16 is a box and whisker plot of the perceived virtual source positions for all participants. For each box, the red line indicates the median value that was identified, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points the algorithm considers not to be outliers, and the red crosses are the outliers. To be considered an outlier, a data point must be more than 1.5 IQR (inter quartile range) below the 25th percentile or above the 75th percentile.

Therefore, the smaller the box, the more concentrated and accurate the responses were to the calculated value. As you can see, the responses around $\pm 45°$ and $0°$ seem to be the most accurate.

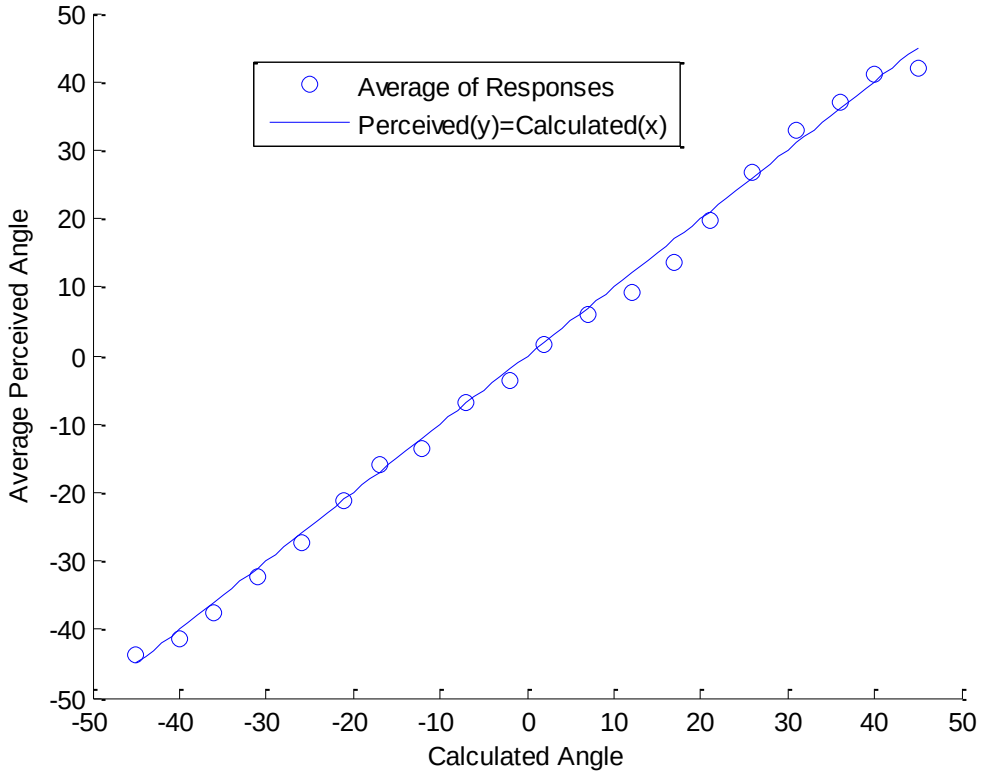

Figure 17: Random Angle Tests – Average of Responses

Figure 17 shows how the average perceived angle from all participants compared to the theoretical values. The line represents the theoretical value, and the circles represent the average of all of the participant's responses. The range of absolute error between the average of the responses and the theoretical angles was from $0.67°$ to $3.53°$

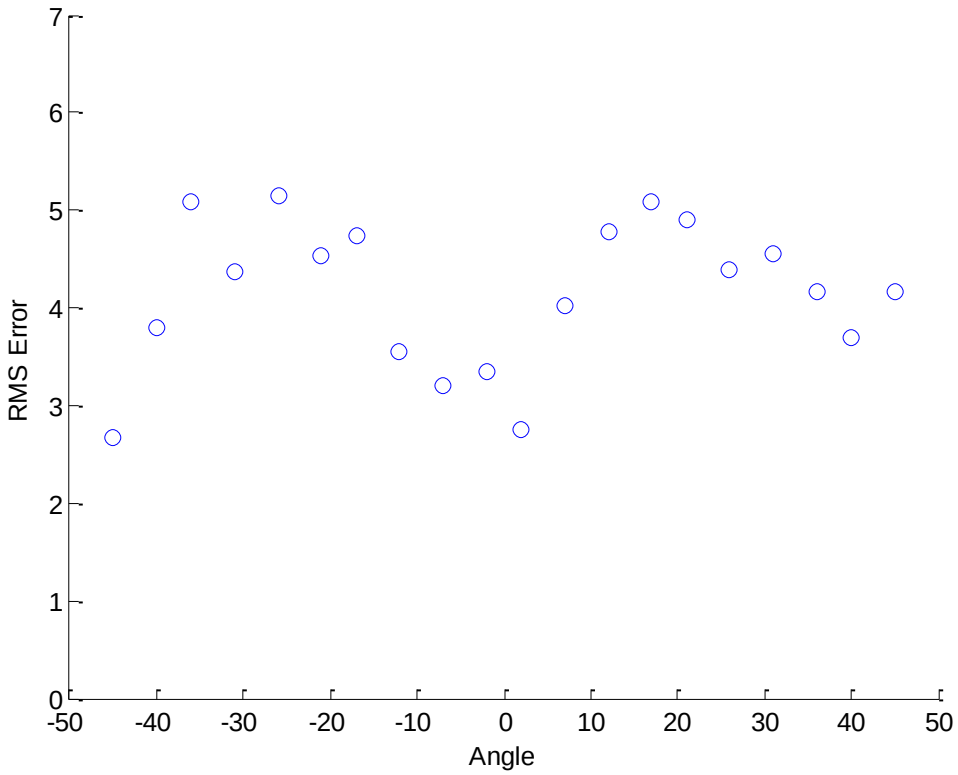

Figure 18: Random Angle Test - RMS error

Figure 18 shows the root mean square error (RMSE) of the average perceived angles from all participants. The equation for the root mean square is

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \mathrm{Error}_i^2}{n}} \qquad (35)$$

where Error is the difference between the perceived and actual angle, and n is the number of participants. The minimum error was at $-45°$ with a RMSE of $2.67°$, and the maximum error was at $17°$ with an RMSE of $5.09°$. As you can see, the region near the center of the arc has the least error, while the regions from $-17°$ to $-36°$ and from $12°$ to $31°$ have the most error.

4.2 Multi-Channel Figures and Summary

In the multi-channel tests, the participant was given a discrete set of points from which to choose the perceived virtual source position (Figure 12 and Figure 13). Each point had an azimuth of $0°, \pm 15°, \pm 30°, \text{or } \pm 45°$ and an elevation of $0°, 14°, 28°, \text{or } 42°$. Therefore, for each point identified there were three scenarios: only the azimuth was correct, only the elevation was correct, or both the azimuth and elevation were correct. Figure 19a, Figure 19b, Figure 20a, and Figure 20b show the results from the multi-channel testing. The first two figures are the Perimeter Test results, and the last two figures are the Random Test results. Figure 19a and Figure 20a are bar plots showing the percentage of responses that matched the exact theoretical value for each virtual source position by all participants. Figure 19b and Figure 20b show the percentage of responses that were within $15°$ of the theoretical value.

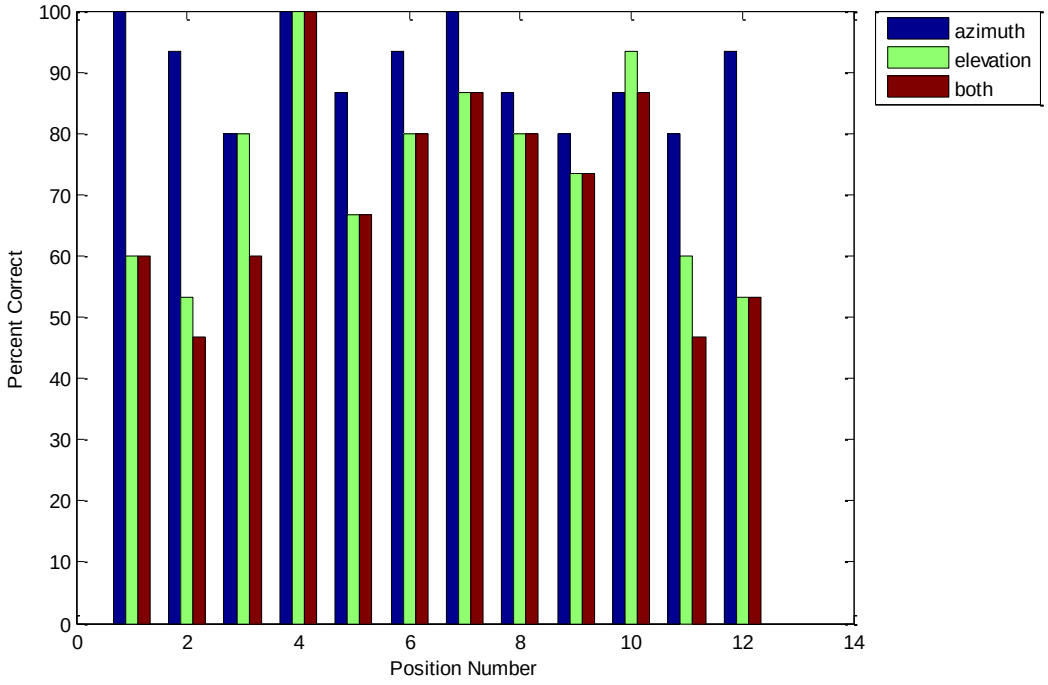

Figure 19a: Perimeter Test results – percent of correct responses

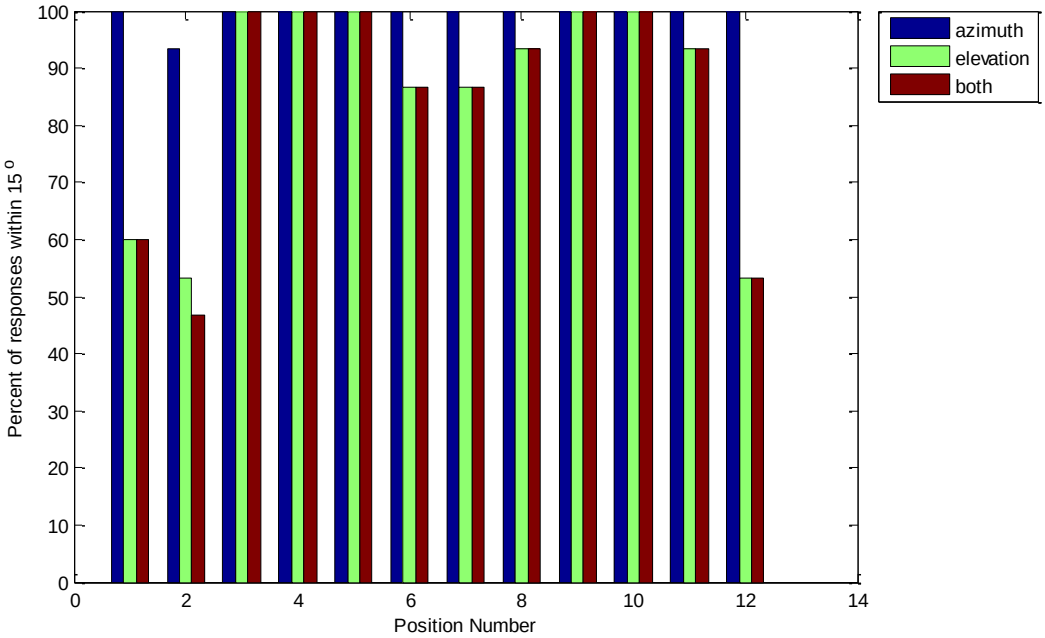

Figure 19b: Perimeter Test results – percent of responses within $15°$

As you can see from Figure 19a, for almost every position participants identified the azimuth with a higher accuracy than the elevation. The responses for azimuth have an accuracy from 80% to 100%, elevation have an accuracy from $53.\overline{3}\%$ to 100%, and both have an accuracy from $46.\overline{6}\%$ to 100%. In Figure 19b there are five positions in which 100% of the responses were within $15°$ of the theoretical value. These five points were all located near the left and right extremes near $\pm 45°$. Only three points out of the twelve have an accuracy of less than $86.\overline{6}\%$. The point with the lowest accuracy is Position 2, which has an azimuth of $15°$ and an elevation of $0°$. For this point, the azimuth was identified correctly $93.\overline{3}\%$ of the time, elevation $53.\overline{3}\%$, and both only $46.\overline{6}\%$. Other points of interest are positions 1, 3, 11, and 12, which have at least one of the categories at 60% or below in the first plot. It's interesting to note that these five points have an azimuth between $-30°$ and $30°$, and an elevation of $0°$.

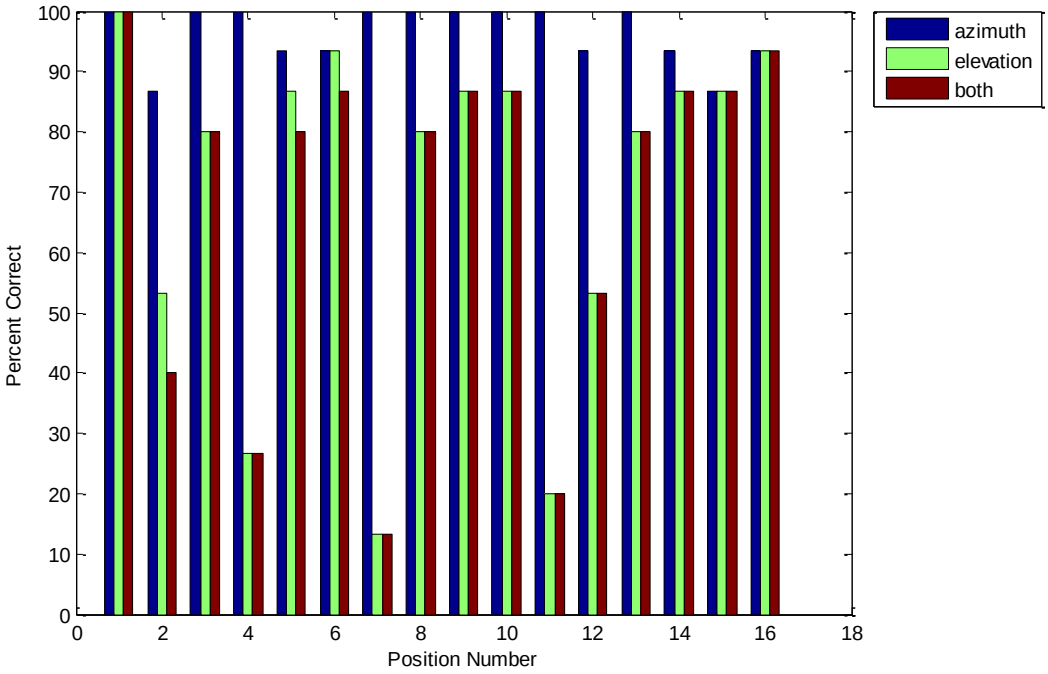

Figure 20a: Random Test results – percent of correct responses

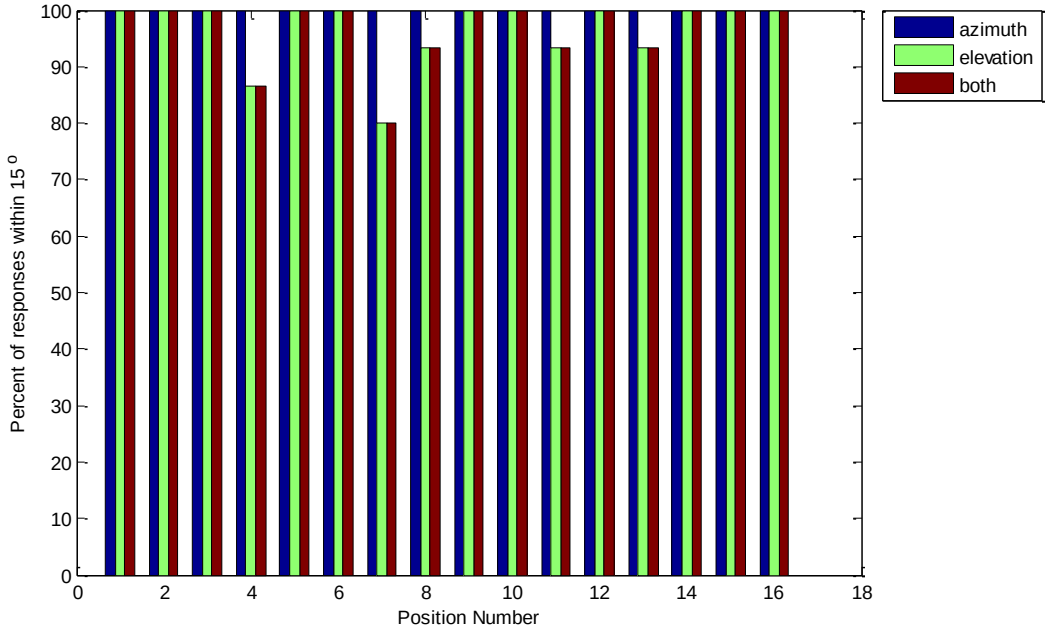

Figure 20b: Random Test results – percent of responses within 15°

The Random Test had similar results to the Perimeter test. Participants identified azimuth with a higher accuracy than elevation for every position. In Figure 20a the responses for azimuth have an accuracy from $86.\overline{6}\%$ to 100%, elevation have an accuracy from $13.\overline{3}\%$ to $93.\overline{3}\%$, and both have an accuracy from $13.\overline{3}\%$ to $93.\overline{3}\%$. In Figure 20b eleven out of the sixteen positions had an accuracy in which 100% of responses were within $15°$ of the theoretical value, and only two of the sixteen points had an accuracy less than $93.\overline{3}\%$. The three least accurate positions are 4, 7, and 11. These points all have an azimuth between $-15°$ and $15°$, and have an elevation of $0°$.

Based on these results, it is easier to identify azimuth than elevation. Intuitively this makes sense due to the location of the ears on the head. Another observation that can be made is that the positions near the x axis at low elevations are the hardest to pinpoint. A possible cause of this is the placement of the speakers. The left and right speakers were positioned to be about the same height as the listener's ears. Because of this, during multi-channel tests, the listener was having to differentiate between four elevations that were all above his/her head. If the bottom speakers were placed lower, the listener would be aligned with the center of the active triangle, and the range of elevations would extend below them as well. Although the experiment could have been set up differently, according to Figure 20b participants identified the correct azimuth and elevation of the virtual source within $15°$ at least 80% of the time.

4.3 Non Equidistant Results

The purpose of the non-equidistant interaural correction algorithm is to calculate the proper gain factors and time delay needed to compensate for interaural differences when the listener is not the same distance from each speaker. Several examples where

the listener and/or speakers are in different positions are presented below. For each example the corrected gain factors, time delay, and color intensity simulation produced by algorithm are shown.

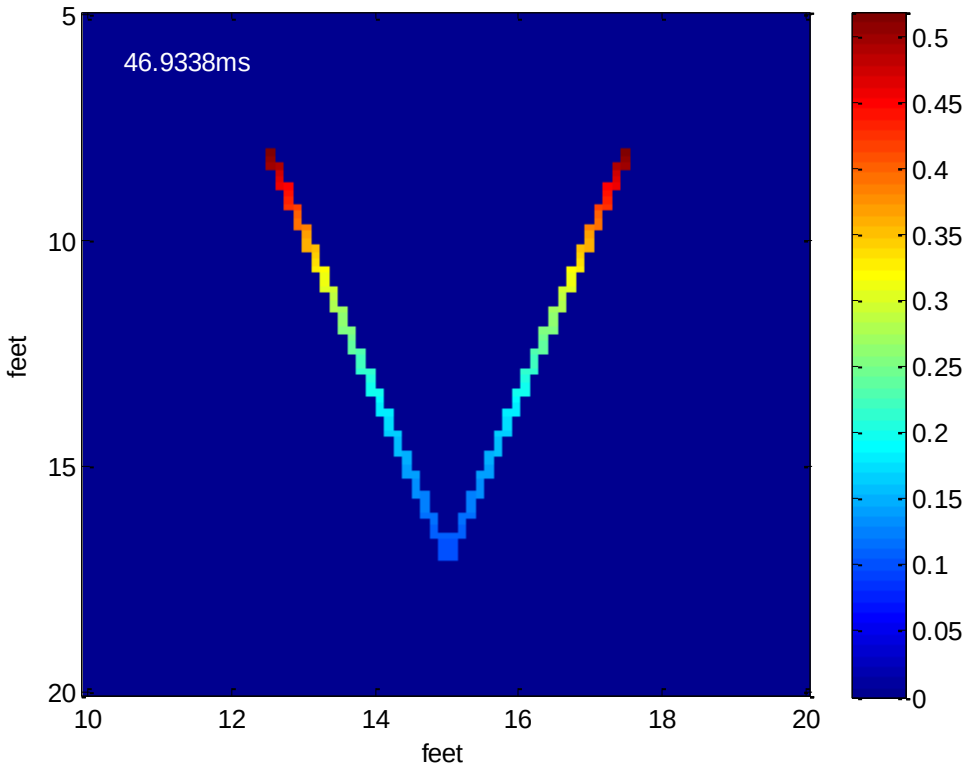

Figure 21a: Interaural Correction Algorithm example 1

Figure 21a shows the scenario in which the listener *is* the same distance from both speakers. The coordinates of the left speaker are (10,20), the right speaker is at (20,20), and the listener is at (15,5). Since the listener is in the middle, the algorithm calculated the corrected gains to be equal: $g_1 = 0.5270 \text{ and } g_2 = 0.5270$, and that there is no time delay: $t_d = 0\, ms$.

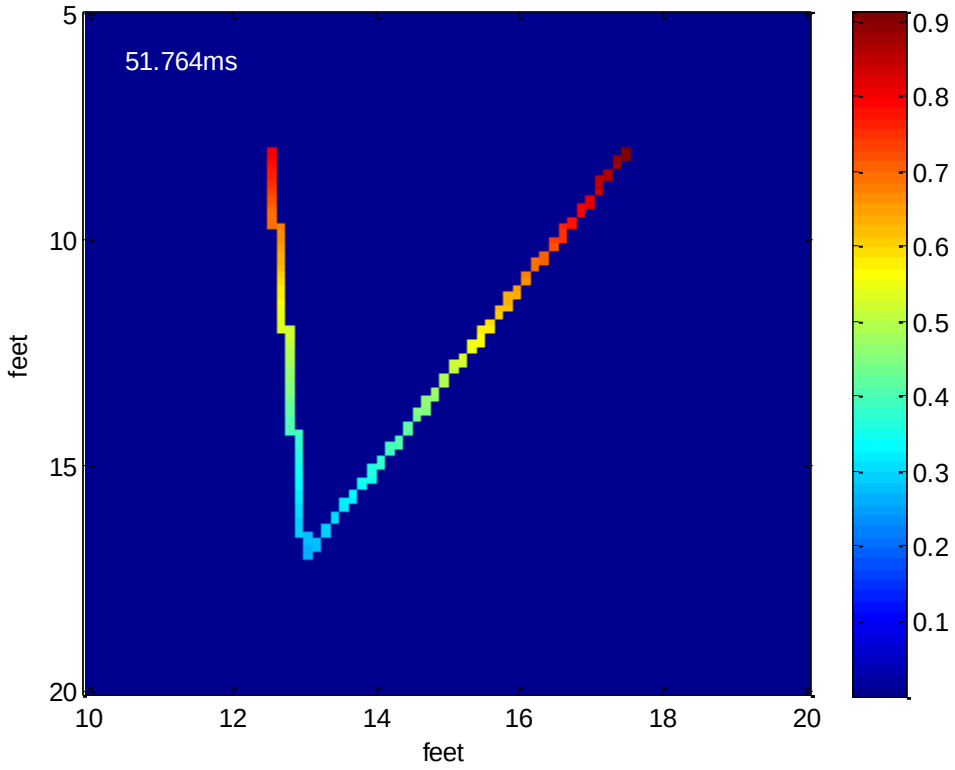

Figure 21b: Interaural Correction Algorithm example 2

In Figure 21b the coordinates of the left speaker are (10,20), the right speaker is at (20,20), and the listener is at (11,5). The interaural correction algorithm calculated the corrected gains to be $g_1 = 0.5421$ and $g_2 = 0.734$, and for the time delay of the left speaker to be $t_d = 7.2278\, ms$.

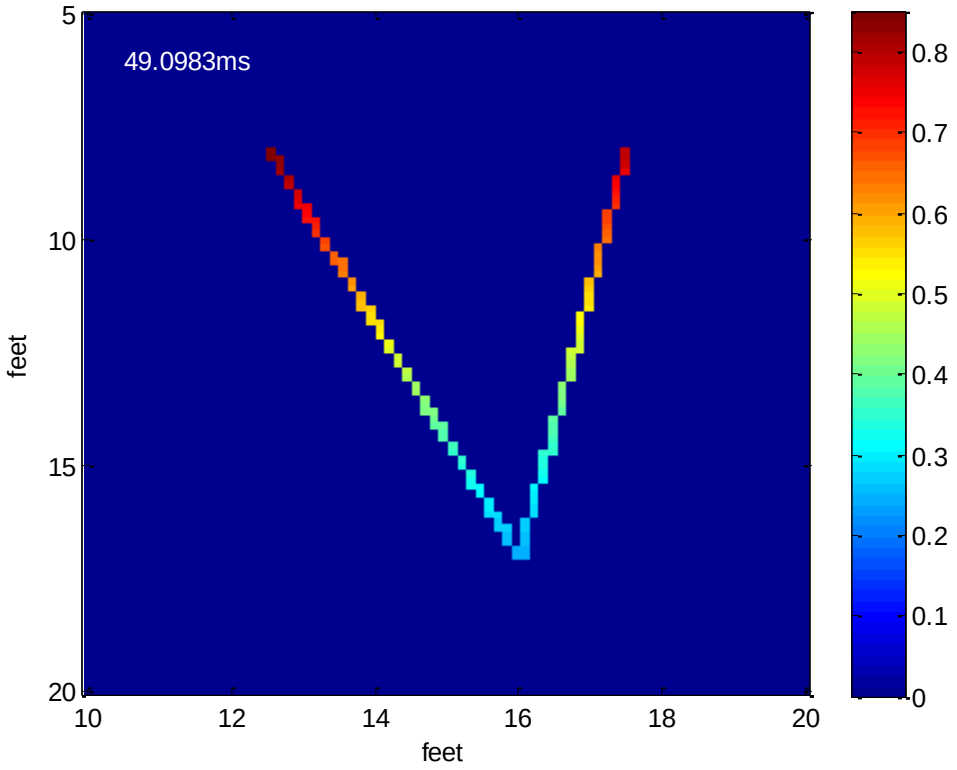

Figure 21c: Interaural Correction Algorithm example 3

In Figure 21c the coordinates of the left speaker are (10,20), the right speaker is at (20,20), and the listener is at (17,5). The interaural correction algorithm calculated the corrected gains to be $g_1 = 0.6216$ and $g_2 = 0.5308$, and for the time delay of the right speaker to be $t_d = 3.6906\, ms$.

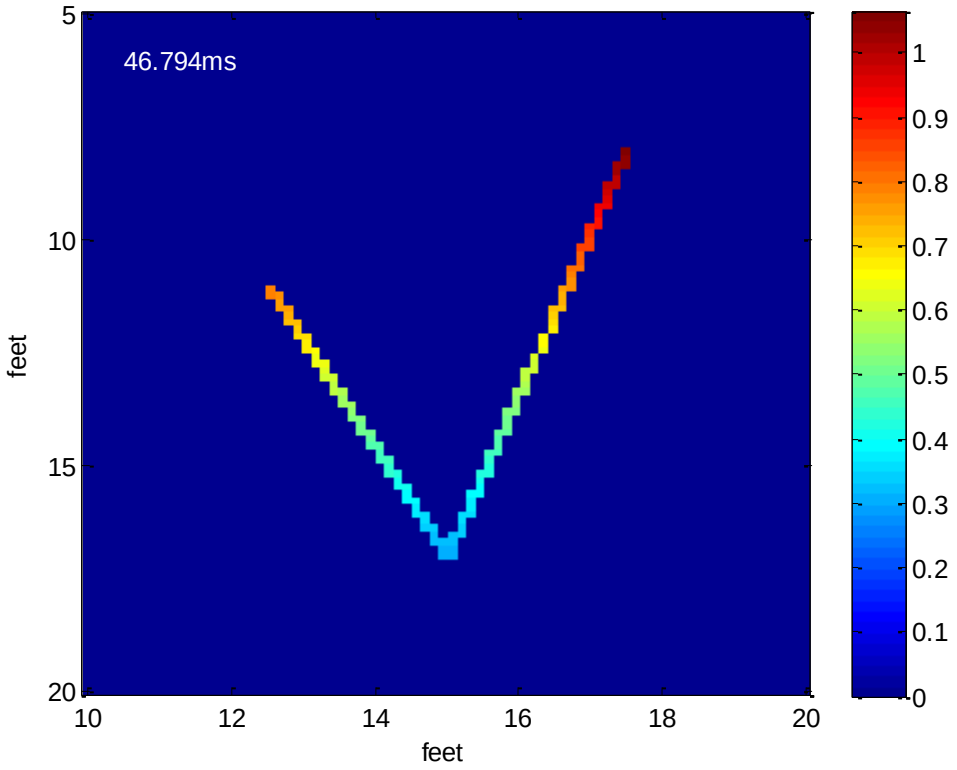

Figure 21d: Interaural Correction Algorithm example 4

In Figure 21d the coordinates of the left speaker are (10,15), the right speaker is at (20,20), and the listener is at (15,5). The interaural correction algorithm calculated the corrected gains to be $g_1 = 0.5398 \text{ and } g_2 = 1.0797$, and for the time delay of the left speaker to be $t_d = 13.609\, ms$.

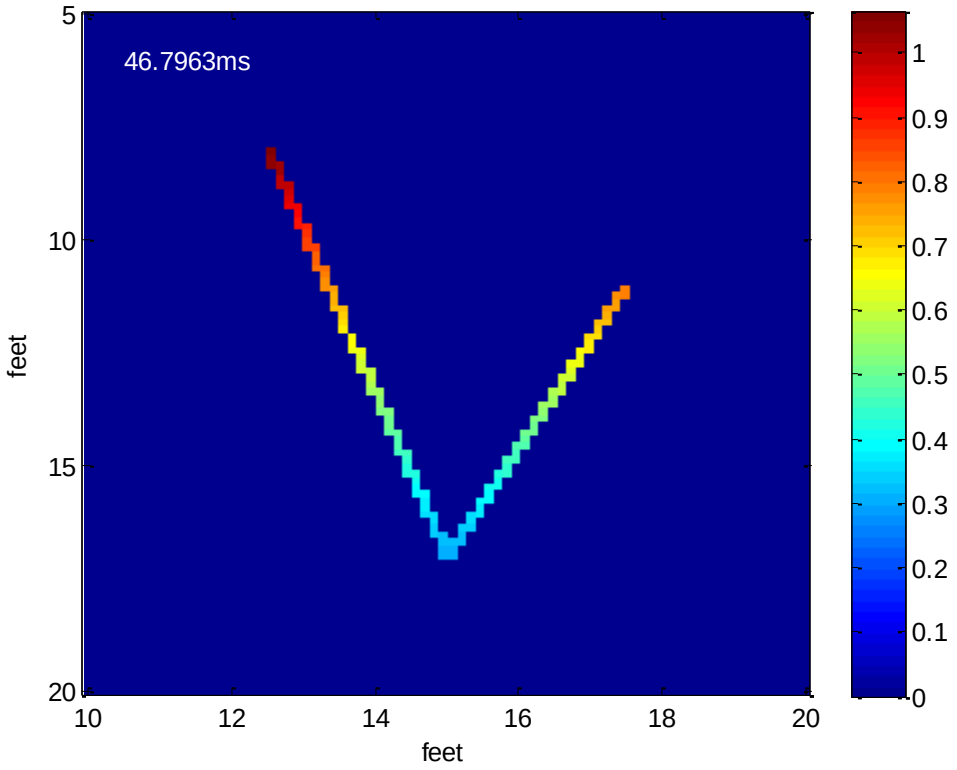

Figure 21e: Interaural correction Algorithm example 5

In Figure 21e the coordinates of the left speaker are (10,15), the right speaker is at (20,20), and the listener is at (15,5). The interaural Correction algorithm calculated the corrected gains to be $g_1 = 1.0797 \text{ and } g_2 = 0.5983$, and for the time delay of the right speaker to be $t_d = 13.609\, ms$.

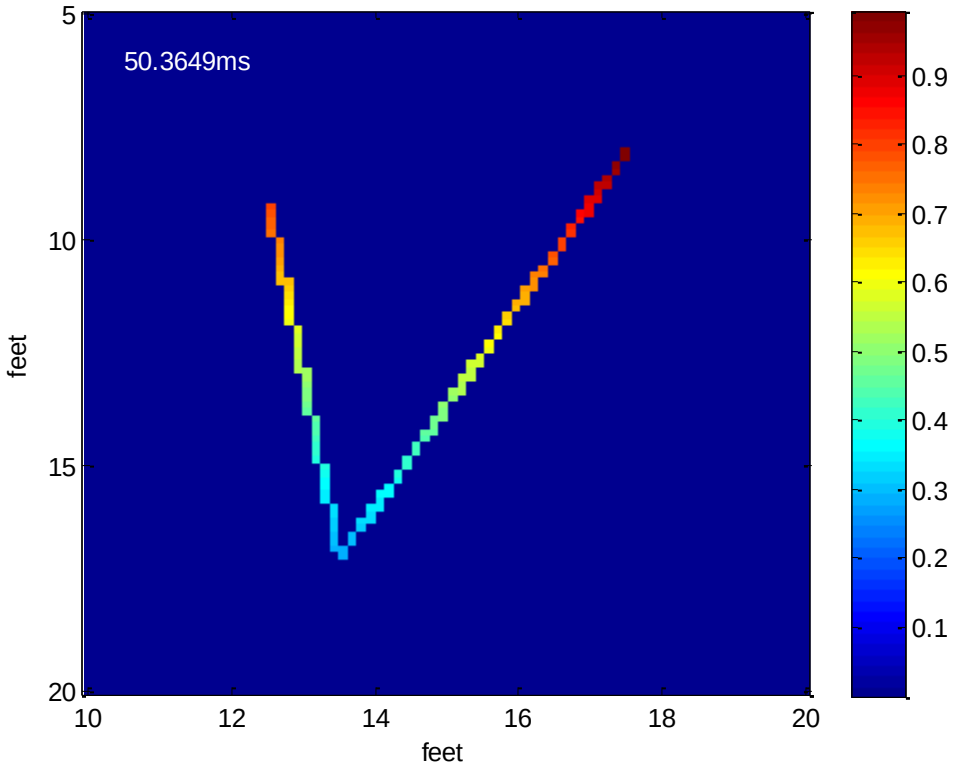

Figure 21f: Interaural Correction Algorithm example 6

In Figure 21f the coordinates of the left speaker are (10,18), the right speaker is at (20,20), and the listener is at (12,5). The interaural correction algorithm calculated the corrected gains to be $g_1 = 0.5384$ and $g_2 = 0.8995$, and for the time delay of the left speaker to be $t_d = 11.305\, ms$.

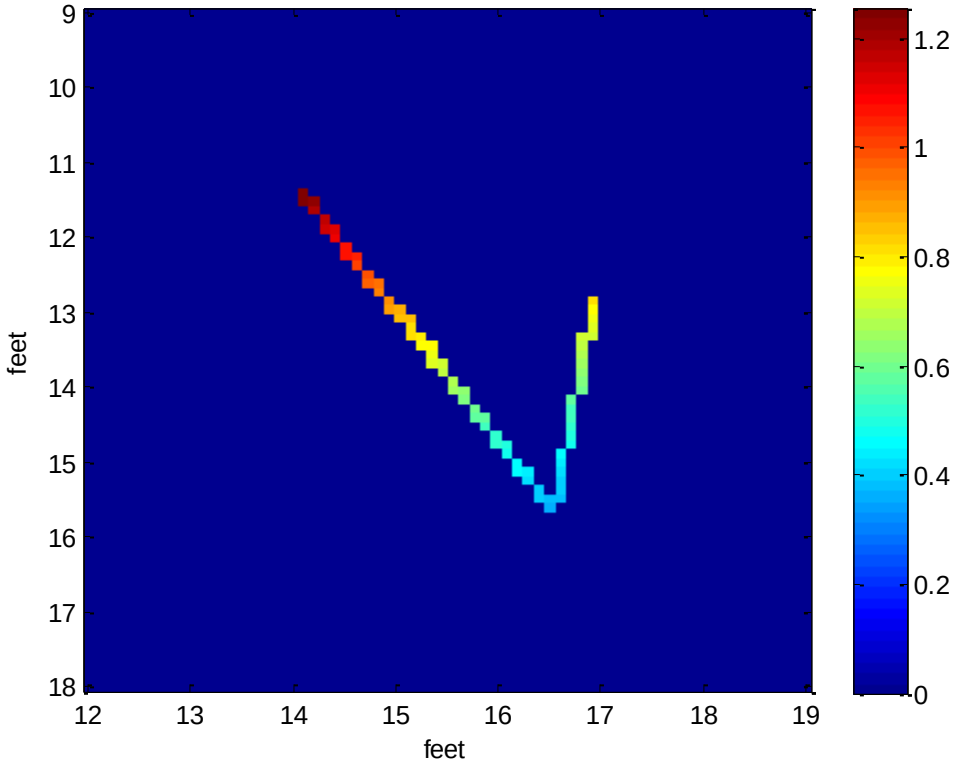

Figure 21g: Interaural Correction Algorithm example 7

In Figure 21g the coordinates of the left speaker are (12,18), the right speaker is at (19,15), and the listener is at (18,9). The interaural correction algorithm calculated the corrected gains to be $g_1 = 1.7813 \text{ and } g_2 = 0.5633$, and for the time delay of the left speaker to be $t_d = 13.911\, ms$.

As you can see, in each example the speaker that is farthest from the observer starts at a higher intensity. Once the preliminary wave reaches the point that is in line with the other speaker, the sound intensities are equal. As the waves move toward the observer both intensities decrease with distance from the source. Since the two lines reach the listener at the same time with the same intensity (color) the sound will be perceived directly in front of them, even though they are closer to one speaker.

CHAPTER 5: CONCLUSION & FUTURE WORK

In this research effort, an algorithm was developed to simulate auditory localization in a two and three channel speaker system. A test setup was produced to implement the simulation, and experimental data was collected to verify the simulation. An interaural correction algorithm was also developed to simulate the scenario in which the listener is not the same distance from each speaker. The 2D experiments showed that participants correctly identified the position of the virtual source with an average RMS error of 4º. The 3D experiments showed that participants identified the correct azimuth and elevation of the virtual source within $15°$ of the simulated position at least 80% of the time. Based on these results the simulation is accurate, and could be useful in implementing a multichannel 3D audio system that could provide a more realistic listening experience than current stereo-sound systems.

Other than positioning the bottom speakers lower, there are several other ways the experiment could be modified and continued. The speakers could be moved to different distances and angles from the listener to see if their ability to distinguish location is affected, or experiments could be done to see how factors like these effect the size of the sweet spot. Other sounds, such as music or speech, could also be used for testing. The next step would be to add additional speakers around the listener to form an “active sphere” in which the sound could be moved. In this scenario, the same math could be used, but the algorithm would have to distinguish which group of three speakers to activate based on the desired virtual source position. The algorithm could also be used to produce a series of locations to play sound over to give it the effect of moving across the active triangle.

The interaural correction algorithm for non-equidistant localization could also be tested. If the experiment matched the simulation, then it could be combined with the 3D amplitude panning algorithm to produce 3D sound despite the listener's location relative to the speakers. Also, if the listener's position were tracked, the algorithm could be modified to update as they move around and place the sound wherever they are.

REFERENCES

[1] Lord Rayleigh [J. W. Strutt], “On our Perception of Sound Direction”, *Philosophical Magazine*, vol. 13, pp. 214-232, 1907.

[2] Jens Blauert, *Spatial Hearing: the Psychophysics of Human Sound Localization*, MIT Press, Cambridge, Mass, 1983.

[3] William Hartmann, “How We Localize Sound”, *Physics Today*, vol. 52, issue 11, pp. 24-29, 1999.

[4] Hans Wallach, Edwin B. Newman, and Mark R. Rosenzweig, “The Precedence Effect in Sound Localization”, *The American Journal of Psychology,* vol. 62, no. 3, pp. 315-336, Jul. 1949

[5] Willard R. Thurlow and Theodore E. Parks, “Precedence-Suppression Effects for Two Click Sources”, *Perceptual and Motor Skills*, vol. 13, pp. 7-12, Aug. 1961.

[6] J.P.A. Lochner and J.F. Burger, “The Subjective Masking of Short Time Delayed Echoes by Their Primary Sounds and Their Contribution to the Intelligibility of Speech”, *Acta Acustica united with Acustica*, vol. 8, no. 1, pp. 1-10, 1958.

[7] Ville Pulkki, “Virtual Sound Source Positioning Using Vector Base Amplitude Panning”, *Journal of the Audio Engineering Society*, vol. 45, Issue 6, pp. 456-466, Jun. 1997.

[8] Sacha Spors, H. Wierstorf, A. Raake, F. Melchior, M. Frank, F. Zotter, "Spatial Sound With Loudspeakers and Its Perception: A Review of the Current State", *Proceedings of the IEEE* , vol. 101, no. 9, pp. 1920-1938, Sept. 2013.

[9] K. Boer, “Plastische Klangwiedergabe”, *Philips Technische Rundschau*, no. 5, pp. 108-115, 1940.

[10] B.B. Bauer, “Stereophonic Earphones and Binaural Loudspeakers”, *Journal of the Audio Eng. Society*, vol. 9, no. 2, pp. 148-151, 1961.

[11] K. Wendt, “Das Richtungshoren bei der Oberlagerung zweier Schallfelder bei Intensitats und Laufzeitstereophonie”, PhD thesis, RWTH Aachen, 1963.

[12] Jeroen Breebaart, Erik Schuijers, “Phantom materialization: A novel method to enhance stereo audio reproduction on headphones”, *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 8, Nov. 2008.

[13] H. A. M. Clark, G. F. Dutton, and P. B. Vanderlyn. The ’Stereosonic’ Recording and Reproduction System: A Two-Channel Systems for Domestic Tape Records”, *J. Audio Eng. Soc.*, vol. 6, Issue 2, pp. 102-117, April 1958.

[14] G. Theile and G. Plenge, “Localization of Lateral Phantom Sources”, *J. Audio Eng. Soc.*, vol. 25, pp. 196–200, 1977.

[15] M.R. Schroeder and B.S. Atal, “Computer Simulation of Sound Transmission in

Rooms”, *Proceedings of the IEEE*, vol. 51, no. 3, pp. 536-537, March 1963.

[16] Myung-Suk Song, Cha Zhang, D. Florencio, Hong-Goo Kang, “An Interactive 3-D Audio System With Loudspeakers”, *Multimedia IEEE Transactions*, vol. 13, no. 5, pp. 844-855, Oct. 2011.

[17] William Gardner, “3-D Audio Using Loudspeakers”, Ph.D. thesis, MIT, MA, 1998.

[18] P. Damaske, “Head-Related Two-Channel Stereophony with Loudspeaker Reproduction”, *J. Acoust. Soc. Am.*, vol. 50, pp. 1109-1115, 1971.

[19] D.H. Cooper and J.L. Bauck 1989, “Prospects for Transaural Recording”, *J. Audio Eng. Soc.*, vol. 37, pp. 3-19, 1989.

[20] J. M. Jot, “Etude et réalisation d'un spatialisateur de sons par modèles physiques et perceptifs”, Doctoral dissertation, Télécom, Paris, France, 1992.

[21] Toshinori Mori and Makoto Iwahara, “Stereophonic Sound Reproduction System”, Patent 04118599, Feb. 1978.

[22] F. L. Wightman and D. J. Kistler, “Headphone Simulation of Free-Field Listening II: Psychophysical validation”, *J. Acoust. Soc. Am.*, vol. 85, pp. 868–878, 1989.

[23] Manabu Okamoto, I. Kinoshita, S. Aoki, H. Matsui, “Sound Image Rendering System for Headphones”, *IEEE Trans. On Consumer Elec.*, vol. 43, Issue 3, pp. 689-693 Aug 1997.

[24] Zhan Huan Zhou, “Sound Localization and Virtual Auditory Space”, University of Toronto, Canada, 2002-2004.

[25] Man-ho Park, Song-in Choi, Si-ho Kim, Keun-Sung Bae, “Improvement of front-back sound localization characteristics in headphone-based 3D sound generation”, *Advanced Communication Technology, ICACT 2005, The 7th International Conference on*, vol. 1, pp. 273-276, 2005.

[26] Yiteng (Arden) Huang. “On Crosstalk Cancellation and Equalization With Multiple Loudspeakers for 3-D Sound Reproduction”, *IEEE Signal Processing Letters*, vol. 14, no. 10, Oct. 2007.

[27] B. Van Daele and W. Van Baelen, "Productions in Auro 11.1", *Barco White Paper,* 2012.

[28] K. Hamasaki, K. Hiyama, and R. Okumura, "The 22.2 Multichannel Sound System and its Application," presented at the 118th Convention of the Audio Engineering Society, paper 6406, May 2005.

APPENDIX A: Experimental Data

Table A1: Two channel experimental data – participants 1 to 5 (all angles in degrees)

			Perceived Angle				
Left Gain	Right Gain	Calculated Angle	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
0.788010754	0.615661475	7	15	8	5	5	12
1	0	45	45	45	42	42	41
0.087155743	0.996194698	-40	-45	-37	-45	-45	-40
0.882947593	0.469471563	17	14	15	7	15	20
0.913545458	0.406736643	21	20	22	17	17	26
0.68199836	0.731353702	-2	-9	-7	-5	-5	-2
0.838670568	0.544639035	12	14	7	5	10	8
0.987688341	0.156434465	36	40	42	32	37	39
0.544639035	0.838670568	-12	-15	-12	-15	-12	-19
0.615661475	0.788010754	-7	-10	-4	-12	-8	-5
0	1	-45	-45	-37	-42	-44	-43
0.325568154	0.945518576	-26	-25	-23	-28	-25	-32
0.970295726	0.241921896	31	35	40	33	35	35
0.731353702	0.68199836	2	7	5	0	0	0
0.469471563	0.882947593	-17	-20	-13	-14	-10	-20
0.406736643	0.913545458	-21	-20	-20	-20	-17	-27
0.945518576	0.325568154	26	35	33	30	30	27
0.156434465	0.987688341	-36	-35	-32	-35	-42	-42
0.241921896	0.970295726	-31	-30	-34	-32	-30	-36
0.996194698	0.087155743	40	40	45	40	43	43

Table A2: Two channel experimental data – participants 6 to 10 (all angles in degrees)

			Perceived Angle				
Left Gain	Right Gain	Calculated Angle	Participant 6	Participant 7	Participant 8	Participant 9	Participant 10
0.788010754	0.615661475	7	4	2	7	5	7
1	0	45	43	40	45	45	38
0.087155743	0.996194698	-40	-35	-40	-42	-45	-45
0.882947593	0.469471563	17	18	10	13	15	15
0.913545458	0.406736643	21	29	20	26	22	15
0.68199836	0.731353702	-2	-3	-3	0	-6	0
0.838670568	0.544639035	12	17	5	14	10	13
0.987688341	0.156434465	36	35	32	42	35	38
0.544639035	0.838670568	-12	-15	-13	-12	-19	-11
0.615661475	0.788010754	-7	-10	-8	-3	-10	-9
0	1	-45	-45	-45	-45	-45	-40
0.325568154	0.945518576	-26	-35	-29	-17	-32	-23
0.970295726	0.241921896	31	32	26	37	30	37
0.731353702	0.68199836	2	2	0	0	0	0
0.469471563	0.882947593	-17	-20	-11	-14	-22	-20
0.406736643	0.913545458	-21	-27	-15	-16	-27	-25
0.945518576	0.325568154	26	25	24	28	20	25
0.156434465	0.987688341	-36	-31	-40	-40	-37	-40
0.241921896	0.970295726	-31	-31	-34	-31	-39	-30
0.996194698	0.087155743	40	43	40	45	36	45

Table A3: Two channel experimental data – participants 11 to 15 (all angles in degrees)

			Perceived Angle				
Left Gain	Right Gain	Calculated Angle	Participant 11	Participant 12	Participant 13	Participant 14	Participant 15
0.788010754	0.615661475	7	9	5	8	0	0
1	0	45	42	38	40	45	37
0.087155743	0.996194698	-40	-36	-44	-44	-38	-39
0.882947593	0.469471563	17	16	12	16	8	8
0.913545458	0.406736643	21	20	15	22	10	17
0.68199836	0.731353702	-2	1	-7	-5	-4	0
0.838670568	0.544639035	12	13	4	8	8	4
0.987688341	0.156434465	36	39	35	44	35	29
0.544639035	0.838670568	-12	-6	-10	-15	-15	-14
0.615661475	0.788010754	-7	-2	-1	-8	-7	-5
0	1	-45	-43	-45	-45	-45	-45
0.325568154	0.945518576	-26	-21	-30	-35	-25	-29
0.970295726	0.241921896	31	30	34	37	26	28
0.731353702	0.68199836	2	8	0	0	0	0
0.469471563	0.882947593	-17	-8	-18	-22	-10	-17
0.406736643	0.913545458	-21	-15	-25	-26	-20	-16
0.945518576	0.325568154	26	24	30	27	23	19
0.156434465	0.987688341	-36	-32	-44	-44	-28	-40
0.241921896	0.970295726	-31	-28	-37	-40	-24	-30
0.996194698	0.087155743	40	35	45	44	36	38

Table A4: Multi-channel Perimeter Test positions (all angles in degrees)

Perimeter Training Positions	Azimuth	Elevation
1	0	0
2	15	0
3	30	0
4	45	0
5	30	14
6	15	28
7	0	42
8	-15	28
9	-30	14
10	-45	0
11	-30	0
12	-15	0

Table A5: Perimeter Test data – participants 1 to 5

	Perceived Position				
Perimeter Test Order	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
2	2	2	3	4	4
3	3	3	3	3	3
4	5	4	4	2	4
9	8	9	9	9	10
7	7	7	6	7	7
0	0	0	0	0	6
8	10	8	8	8	8
6	6	6	6	6	6
1	5	1	3	5	5
5	1	5	5	5	5
11	11	11	7	11	7
10	8	10	8	10	8

Table A6: Perimeter Test data – participants 6 to 10

	Perceived Position				
Perimeter Test Order	Participant 6	Participant 7	Participant 8	Participant 9	Participant 10
2	2	2	4	2	2
3	3	3	3	3	3
4	4	4	2	4	4
9	9	9	9	9	9
7	7	7	7	7	7
0	0	6	0	0	0
8	8	8	8	8	8
6	6	0	6	6	6
1	1	5	5	1	1
5	4	5	1	5	5
11	7	7	11	11	11
10	9	10	10	9	8

Table A7: Perimeter Test data – participants 11 to 15

	Peceived Position				
Perimeter Test Order	Participant 11	Participant 12	Participant 13	Participant 14	Participant 15
2	2	2	3	2	2
3	3	3	3	3	3
4	2	4	4	4	4
9	9	9	9	9	9
7	7	7	7	7	7
0	0	6	6	6	0
8	8	9	9	8	8
6	6	0	6	6	6
1	1	5	1	1	1
5	5	5	5	5	5
11	7	11	7	11	11
10	10	10	8	10	10

Table A8: Multi-channel Random Angle Test positions (all angles in degrees)

Random Test Training Positions	Azimuth	Elevation
1	45	0
2	30	0
3	30	14
4	15	0
5	15	14
6	15	28
7	0	0
8	0	14
9	0	28
10	0	42
11	-15	0
12	-15	14
13	-15	28
14	-30	0
15	-30	14
16	-45	0

Table A9: Random Angle Test data – participants 1 to 5

	Perceived Position				
Random Test Order	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
10	10	10	9	10	10
13	12	12	13	13	13
9	8	9	10	9	9
4	5	5	6	6	5
6	6	6	9	5	6
2	1	2	2	3	2
15	15	16	15	15	16
14	14	14	14	14	14
8	8	8	10	8	9
5	5	4	5	5	5
11	12	11	12	12	12
7	9	7	8	8	8
1	1	1	1	1	1
16	16	16	16	16	16
3	3	2	3	3	3
12	12	12	13	11	12

Table A10: Random Angle Test data – participants 5 to 10

	Perceived Position				
Random Test Order	Participant 6	Participant 7	Participant 8	Participant 9	Participant 10
10	10	10	10	10	10
13	13	13	12	13	13
9	9	9	9	9	9
4	5	4	4	5	4
6	6	6	6	6	6
2	2	3	3	3	1
15	15	15	15	15	15
14	14	14	14	14	14
8	8	8	8	8	8
5	5	5	5	5	5
11	11	12	12	12	12
7	9	8	8	8	8
1	1	1	1	1	1
16	16	16	16	16	16
3	2	3	3	3	3
12	12	12	12	12	12

Table A11: Random Angle Test data – participants 11 to 15

	Perceived Position				
Random Test Order	Participant 11	Participant 12	Participant 13	Participant 14	Participant 15
10	10	9	10	10	10
13	13	13	13	13	13
9	9	9	9	9	9
4	5	4	5	5	5
6	6	6	6	6	6
2	2	3	3	3	2
15	15	15	15	15	15
14	14	15	12	14	14
8	8	7	8	8	8
5	5	5	6	5	5
11	12	13	12	11	12
7	8	7	8	9	8
1	1	1	1	1	1
16	16	16	15	16	16
3	3	3	2	3	3
12	13	13	13	12	13

APPENDIX B: MATLAB Code

TrainingProgram2D.m

```
clear all
close all
choosetest = mymenu1('Choose Training:','Sweep','User Input');
if choosetest == 1
    % % get speaker inputs
    prompt = {'Please enter the distance to the speakers in feet:
',...
        'Please enter the angle (in degrees) from center to the left
speaker: ',...
        'Please enter the angle (in degrees) from center to the right
speaker: '};

    title = 'Speaker Parameters';
    numlines = 1;
    defAns = {'8','45','-45'};
    options.Resize='off';
    options.WindowStyle='normal';
    options.Interpreter='tex';

    answer = inputdlg(prompt,title,numlines,defAns,options);
    s = str2num(answer{1});
    LeftAngle = str2num(answer{2});
    RightAngle = str2num(answer{3});

    % Calculate angles and gains
    for Pangle = RightAngle:15:LeftAngle  % for stepping through all
angles

        xL = s*cosd(LeftAngle);
        yL = s*sind(LeftAngle);
        Left = [xL yL];
        xR = s*cosd(RightAngle);
        yR = s*sind(RightAngle);
        Right = [xR yR];
        xP = s*cosd(Pangle);
        yP = s*sind(Pangle);
        P = [xP yP];

        done = false;
        while (~done)
            LeftMag = norm(Left);
```

```
            if Left(1)>0 && Left(2)>0
                done = true;
            elseif ~(Left(1)>0 && Left(2)>0)
                fprintf('Error: Speaker out of bounds, must be in
first quadrant \n');
                Left = input('Please input a 2 dimensional vector ([x
y]) for the left speaker: ');
            end
        end
        thetaL = (atan(Left(2)/Left(1)))*180/pi

        done = false;
        while (~done)
            RightMag = norm(Right);
            if Right(1)>0 && Right(2)<0 && (abs(LeftMag-
RightMag))<10^-6
                done = true;
            elseif ~(Right(1)>0 && Right(2)<0)
                fprintf('Error: Speaker out of bounds, must be in
fourth quadrant \n');
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            elseif ~((abs(LeftMag-RightMag))<10^-6 )
                fprintf('Error: Magnitude of vectors must be equal
\n')
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            end
        end
        thetaR = (atan(Right(2)/Right(1)))*180/pi

        done = false;
        while (~done)
            P = P';
            thetaP = (atan(P(2)/P(1)))*180/pi
            L = [Left' Right']';
            G = P'*inv(L)
            if P(1)>0 && Pangle>=RightAngle && Pangle<=LeftAngle &&
max(G)<=1 && max(G)>=0
                done = true;
            elseif ~(P(1)>0 && Pangle>RightAngle && Pangle<LeftAngle
&& max(G)<1 && max(G)>0)
                fprintf('Error')
                done = true;
            end
        end
```

```
        gL = G(1);
        gR = G(2);

        perceivedL = sin(2*pi*400*(0:1/20000:2));
        perceivedR = sin(2*pi*400*(0:1/20000:2));
        perceived = [gL.*perceivedL' gR.*perceivedR'];

        %% Play tone
        soundsc(perceived,20000);
        pause(2.5)
    end

else
    trainingdone = 2;
    while trainingdone == 2
        close all
        %% Play tone at 0,+-15,+-30,,+-45

        prompt = {'Please enter the distance to the speakers in feet:
',...
            'Please enter the angle (in degrees) from center to the
left speaker: ',...
            'Please enter the angle (in degrees) from center to the
right speaker: ',...
            'Please enter the desired angle (in degrees) of the
perceived sound: '};

        title = 'Speaker Parameters';
        numlines = 1;
        defAns = {'8','45','-45','0'};
        options.Resize='off';
        options.WindowStyle='normal';
        options.Interpreter='tex';

        answer = inputdlg(prompt,title,numlines,defAns,options);
        s = str2num(answer{1});
        LeftAngle = str2num(answer{2});
        RightAngle = str2num(answer{3});
        Pangle = str2num(answer{4});
        xL = s*cosd(LeftAngle);
        yL = s*sind(LeftAngle);
        Left = [xL yL];
        xR = s*cosd(RightAngle);
        yR = s*sind(RightAngle);
        Right = [xR yR];
        xP = s*cosd(Pangle);
        yP = s*sind(Pangle);
```

```
        P = [xP yP];

        done = false;
        while (~done)
            LeftMag = norm(Left);
            if Left(1)>0 && Left(2)>0
                done = true;
            elseif ~(Left(1)>0 && Left(2)>0)
                fprintf('Error: Speaker out of bounds, must be in
first quadrant \n');
                Left = input('Please input a 2 dimensional vector ([x
y]) for the left speaker: ');
            end
        end
        thetaL = (atan(Left(2)/Left(1)))*180/pi

        done = false;
        while (~done)
            RightMag = norm(Right);
            if Right(1)>0 && Right(2)<0 && (abs(LeftMag-
RightMag))<10^-6
                done = true;
            elseif ~(Right(1)>0 && Right(2)<0)
                fprintf('Error: Speaker out of bounds, must be in
fourth quadrant \n');
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            elseif ~((abs(LeftMag-RightMag))<10^-6 )
                fprintf('Error: Magnitude of vectors must be equal
\n')
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            end
        end
        thetaR = (atan(Right(2)/Right(1)))*180/pi

        done = false;
        while (~done)
            P = P';
            thetaP = (atan(P(2)/P(1)))*180/pi
            L = [Left' Right'];
            G = P'*inv(L)
            if P(1)>0 && Pangle>=RightAngle && Pangle<=LeftAngle &&
max(G)<=1 && max(G)>=0
                done = true;
            elseif ~(P(1)>0 && Pangle>RightAngle && Pangle<LeftAngle
&& max(G)<1 && max(G)>0)
```

```
                prompt = {'Error: The Perceived sound must lie in the
active arc of the two speakers   Please enter the desired angle of the
perceived sound : '};
                answer = inputdlg(prompt);
                Pangle = str2num(answer{1});
                xP = s*cosd(Pangle);
                yP = s*sind(Pangle);
                P = [xP yP];
            end
        end

        figure
        plot([0 Left(2)],[0 Left(1)],'r-o')
        hold on
        plot([0 Right(2)],[0 Right(1)],'-o')
        plot([0 P(2)],[0 P(1)],'k-x')
        set(gca,'XDir','reverse')
        legend('Left Speaker','Right Speaker','Virtual Source')
        xlabel('feet')
        ylabel('feet')
        grid on

        gL = G(1);
        gR = G(2);

        perceivedL = sin(2*pi*400*(0:1/44100:2));
        perceivedR = sin(2*pi*400*(0:1/44100:2));
        perceived = [gL.*perceivedL' gR.*perceivedR'];

        soundsc(perceived,44100);

        repdone = false;
        while (~repdone)
            replay = mymenu1('Replay Sound?','Yes','No');
            if replay ==1
                soundsc(perceived,20000);
            else
                repdone=true;
            end
        end
        trainingdone = mymenu1('Training Done?','Yes','No');
    end
end
```

DiscreteAngleTest2D.m

```
clear all
close all

prompt = {'Please enter the Participant number: ',...
    'Please enter the frequency of the tone you want to play: '};

title = 'Participant Information';
numlines = 1;
defAns = {'','400'};
options.Resize='off';
options.WindowStyle='normal';
options.Interpreter='tex';
answer = inputdlg(prompt,title,numlines,defAns,options);
freq = str2num(answer{2});
Participant = str2num(answer{1});
filename = ['Amplitude Panning Data_' date];

if (~exist([filename '.xls']))
    xlswrite(filename,[{'Trial'} {'Left Gain'}...
        {'Right Gain'} {'Desired Perceived Angle'} {'Actual Perceived
Angle'}...
        {'Error'} {'Distance to Speakers'}],...
        Participant,[char(65) num2str(1)]);
    Trial = 1;
else
    xlswrite(filename,[{'Trial'} {'Left Gain'}...
        {'Right Gain'} {'Desired Perceived Angle'} {'Actual Perceived
Angle'}...
        {'Error'} {'Distance to Speakers'}],...
        Participant,[char(65) num2str(1)]);
    A = xlsread(filename,Participant);
    [rows cols] = size(A);
    Trial = rows+1;
end

testdone = false;
% First 20 only 0:15:+-45    Last 20 use Random
prompt = {'Please enter the distance to the speakers in feet: ',...
    'Please enter the angle (in degrees) from center to the left
speaker: ',...
    'Please enter the angle (in degrees) from center to the right
speaker: '};

title = 'Speaker Parameters';
numlines = 1;
```

```
defAns = {'8','45','-45'};
options.Resize='off';
options.WindowStyle='normal';
options.Interpreter='tex';

answer = inputdlg(prompt,title,numlines,defAns,options);
s = str2num(answer{1});
LeftAngle = str2num(answer{2});
RightAngle = str2num(answer{3});

Angles = [15,-30,30,-15,45,0,-45,-30,-30,45,-15,30,45,-15,0,15,45,-
30,15,-45];

while testdone~=Angles(length(Angles))
    for i = 1:length(Angles);
        close all
        Pangle = Angles(i);
        xL = s*cosd(LeftAngle);
        yL = s*sind(LeftAngle);
        Left = [xL yL];
        xR = s*cosd(RightAngle);
        yR = s*sind(RightAngle);
        Right = [xR yR];
        xP = s*cosd(Pangle);
        yP = s*sind(Pangle);
        P = [xP yP];

        done = false;
        while (~done)
            LeftMag = norm(Left);
            if Left(1)>0 && Left(2)>0
                done = true;
            elseif ~(Left(1)>0 && Left(2)>0)
                fprintf('Error: Speaker out of bounds, must be in
first quadrant \n');
                Left = input('Please input a 2 dimensional vector ([x
y]) for the left speaker: ');
            end
        end
        thetaL = (atan(Left(2)/Left(1)))*180/pi

        done = false;
        while (~done)
            RightMag = norm(Right);
            if Right(1)>0 && Right(2)<0 && (abs(LeftMag-
RightMag))<10^-6
                done = true;
```

```
            elseif ~(Right(1)>0 && Right(2)<0)
                fprintf('Error: Speaker out of bounds, must be in
fourth quadrant \n');
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            elseif ~((abs(LeftMag-RightMag))<10^-6 )
                fprintf('Error: Magnitude of vectors must be equal
\n')
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            end
        end
        thetaR = (atan(Right(2)/Right(1)))*180/pi

        done = false;
        while (~done)
            P = P';
            thetaP = (atan(P(2)/P(1)))*180/pi
            L = [Left' Right'];
            G = P'*inv(L)
            if P(1)>0 && Pangle>=RightAngle && Pangle<=LeftAngle &&
max(G)<=1 && max(G)>=0
                done = true;
            elseif ~(P(1)>0 && Pangle>RightAngle && Pangle<LeftAngle
&& max(G)<1 && max(G)>0)
                prompt = {'Error: The Perceived sound must lie in the
active arc of the two speakers   Please enter the desired angle of the
perceived sound : '};
                answer = inputdlg(prompt);
                Pangle = str2num(answer{1});
                xP = s*cosd(Pangle);
                yP = s*sind(Pangle);
                P = [xP yP];
            end
        end

        figure
        plot([0 Left(1)],[0 Left(2)],'r-o')
        hold on
        plot([0 Right(1)],[0 Right(2)],'-o')
        plot([0 P(1)],[0 P(2)],'k-x')
        grid on

        gL = G(1);
        gR = G(2);

        perceivedL = sin(2*pi*freq*(0:1/20000:2));
```

```
        perceivedR = sin(2*pi*freq*(0:1/20000:2));
        perceived = [gL.*perceivedL' gR.*perceivedR'];
        soundsc(perceived,20000);

        repdone = false;
        while (~repdone)
            replay = mymenu1('Replay Sound?','Yes','No');
            if replay ==1
                soundsc(perceived,20000);
            else
                repdone=true;
            end
        end

        prompt2 = {'What was the angle of the perceived sound?: '};
        answer2 = inputdlg(prompt2);
        PerceivedAngle = str2num(answer2{1});
        Error = Pangle-PerceivedAngle;

        row = Trial+1;
        letter = char(65);
        number = num2str(row);
        cell = [letter number];

        xlswrite(filename,[Trial gL gR Pangle PerceivedAngle Error
s],Participant,cell);

        testcomplete = mymenu1('Continue Testing?','Yes','No');
        testdone = Pangle;
        Trial = Trial+1;
    end
end
```

RandomAngleTest2D.m

```
clear all
close all

testdone = false;
prompt = {'Please enter the Participant number: ',...
    'Please enter the frequency of the tone you want to play: '};

title = 'Participant Information';
numlines = 1;
defAns = {'','400'};
options.Resize='off';
options.WindowStyle='normal';
options.Interpreter='tex';
answer = inputdlg(prompt,title,numlines,defAns,options);
freq = str2num(answer{2});
Participant = str2num(answer{1});
filename = ['Amplitude Panning Data_' date];

if (~exist([filename '.xls']))
    xlswrite(filename,[{'Trial'} {'Left Gain'}...
        {'Right Gain'} {'Desired Perceived Angle'} {'Actual Perceived
Angle'}...
        {'Error'} {'Distance to Speakers'}],...
        Participant,[char(65) num2str(1)]);
    Trial = 1;
else
    xlswrite(filename,[{'Trial'} {'Left Gain'}...
        {'Right Gain'} {'Desired Perceived Angle'} {'Actual Perceived
Angle'}...
        {'Error'} {'Distance to Speakers'}],...
        Participant,[char(65) num2str(1)]);
    A = xlsread(filename,Participant);
    [rows cols] = size(A);
    Trial = rows+1;
end

% First 20 only 0:15:+-45    Last 20 Random
prompt = {'Please enter the distance to the speakers in feet: ',...
    'Please enter the angle (in degrees) from center to the left
speaker: ',...
    'Please enter the angle (in degrees) from center to the right
speaker: '};

title = 'Speaker Parameters';
numlines = 1;
```

```
defAns = {'8','45','-45'};
options.Resize='off';
options.WindowStyle='normal';
options.Interpreter='tex';

answer = inputdlg(prompt,title,numlines,defAns,options);
s = str2num(answer{1});
LeftAngle = str2num(answer{2});
RightAngle = str2num(answer{3});
Angles = [7,45,-40,17,21,-2,12,36,-12,-7,-45,-26,31,2,-17,-21,26,-36,-
31,40];

while testdone~=Angles(length(Angles))
for i = 1:length(Angles);

        close all
        Pangle = Angles(i);

        xL = s*cosd(LeftAngle);
        yL = s*sind(LeftAngle);
        Left = [xL yL];
        xR = s*cosd(RightAngle);
        yR = s*sind(RightAngle);
        Right = [xR yR];
        xP = s*cosd(Pangle);
        yP = s*sind(Pangle);
        P = [xP yP];

        done = false;
        while (~done)
            LeftMag = norm(Left);
            if Left(1)>0 && Left(2)>0
                done = true;
            elseif ~(Left(1)>0 && Left(2)>0)
                fprintf('Error: Speaker out of bounds, must be in
first quadrant \n');
                Left = input('Please input a 2 dimensional vector ([x
y]) for the left speaker: ');
            end
        end
        thetaL = (atan(Left(2)/Left(1)))*180/pi

        done = false;
        while (~done)
            RightMag = norm(Right);
            if Right(1)>0 && Right(2)<0 && (abs(LeftMag-
RightMag))<10^-6
```

```
                done = true;
            elseif ~(Right(1)>0 && Right(2)<0)
                fprintf('Error: Speaker out of bounds, must be in
fourth quadrant \n');
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            elseif ~((abs(LeftMag-RightMag))<10^-6 )
                fprintf('Error: Magnitude of vectors must be equal
\n')
                Right = input('Please input a 2 dimensional vector ([x
y]) for the right speaker: ');
            end
        end
        thetaR = (atan(Right(2)/Right(1)))*180/pi

        done = false;
        while (~done)
            P = P';
            thetaP = (atan(P(2)/P(1)))*180/pi
            L = [Left' Right'];
            G = P'*inv(L)
            if P(1)>0 && Pangle>=RightAngle && Pangle<=LeftAngle &&
max(G)<=1 && max(G)>=0
                done = true;
            elseif ~(P(1)>0 && Pangle>RightAngle && Pangle<LeftAngle
&& max(G)<1 && max(G)>0)
                prompt = {'Error: The Perceived sound must lie in the
active arc of the two speakers   Please enter the desired angle of the
perceived sound : '};
                answer = inputdlg(prompt);
                Pangle = str2num(answer{1});
                xP = s*cosd(Pangle);
                yP = s*sind(Pangle);
                P = [xP yP];
            end
        end

        figure
        plot([0 Left(1)],[0 Left(2)],'r-o')
        hold on
        plot([0 Right(1)],[0 Right(2)],'-o')
        plot([0 P(1)],[0 P(2)],'k-x')
        grid on

        gL = G(1);
        gR = G(2);
```

```
        perceivedL = sin(2*pi*freq*(0:1/20000:2));
        perceivedR = sin(2*pi*freq*(0:1/20000:2));
        perceived = [gL.*perceivedL' gR.*perceivedR'];
        soundsc(perceived,20000);

        repdone = false;
        while (~repdone)
            replay = mymenu1('Replay Sound?','Yes','No');
            if replay ==1
                soundsc(perceived,20000);
            else
                repdone=true;
            end
        end

        prompt2 = {'What was the angle of the perceived sound?: '};
        answer2 = inputdlg(prompt2);
        PerceivedAngle = str2num(answer2{1});
        Error = Pangle-PerceivedAngle;

        row = Trial+1;
        letter = char(65);
        number = num2str(row);
        cell = [letter number];

        xlswrite(filename,[Trial gL gR Pangle PerceivedAngle Error
s],Participant,cell);

        testcomplete = mymenu1('Continue Testing?','Yes','No');
        testdone = Pangle;
        Trial = Trial+1;
    end
end
```

GainCalc3D.m

```
close all
clear all
prompt = {'Please enter the distance to the speakers in feet: ',...
    'Please enter the angle (in degrees) from center to the left
speaker: ',...
    'Please enter the angle (in degrees) from center to the right
speaker: ',...
    'Please enter the elevation (in degrees) from horizontal to the
top speaker: ',...
    'Please enter the desired elevation (in degrees) from horizontal
of the perceived sound: ',...
    'Please enter the desired angle (in degrees) of the perceived
sound: '};

title = 'Speaker Parameters';
numlines = 1;
defAns = {'8','45','-45','42','28','15'};
options.Resize='off';
options.WindowStyle='normal';
options.Interpreter='tex';
answer = inputdlg(prompt,title,numlines,defAns,options);
s = str2num(answer{1});
LeftAngle = str2num(answer{2});
RightAngle = str2num(answer{3});
TopElevation = str2num(answer{4});
Pelevation = str2num(answer{5});
Pangle = str2num(answer{6});

% angle = randi([RightAngle LeftAngle])*pi/180; % for random angle
generation
xL = s*cosd(LeftAngle);
yL = s*sind(LeftAngle);
zL = 0;
Left = [xL yL zL];
xR = s*cosd(RightAngle);
yR = s*sind(RightAngle);
zR = 0;
Right = [xR yR zR];
xT = 0;
yT = 0;
zT = s*sind(TopElevation);
xT = sqrt(s^2-zT^2);
yT = 0;
Top = [xT yT zT];
xP = s*cosd(Pangle);
```

```
yP = s*sind(Pangle);
zP = sqrt(xP^2+yP^2)*tand(Pelevation);
P = [xP yP zP];

done = false;
while (~done)
    LeftMag = norm(Left);
    if Left(1)>0 && Left(2)>0
        done = true;
    elseif ~(Left(1)>0 && Left(2)>0)
        fprintf('Error: Speaker out of bounds, must be in first
quadrant \n');
        Left = input('Please input a 2 dimensional vector ([x y]) for
the left speaker: ');
    end
end
thetaL = (atan(Left(2)/Left(1)))*180/pi

done = false;
while (~done)
    RightMag = norm(Right);
    if Right(1)>0 && Right(2)<0 && (abs(LeftMag-RightMag))<10^-6
        done = true;
    elseif ~(Right(1)>0 && Right(2)<0)
        fprintf('Error: Speaker out of bounds, must be in fourth
quadrant \n');
        Right = input('Please input a 2 dimensional vector ([x y]) for
the right speaker: ');
    elseif ~((abs(LeftMag-RightMag))<10^-6 )
        fprintf('Error: Magnitude of vectors must be equal \n')
        Right = input('Please input a 2 dimensional vector ([x y]) for
the right speaker: ');
    end
end
thetaR = (atan(Right(2)/Right(1)))*180/pi

done = false;
while (~done)
    P = P';
    thetaP = (atan(P(2)/P(1)))*180/pi
    elevP = atand(P(3)/P(1))
    L = [Left' Right' Top']';
    G = P'*inv(L)
    if max(G)>1
    G = G/max(G)
    end
```

```
    if P(1)>0 && Pangle>=RightAngle && Pangle<=LeftAngle && max(G)<=1
&& max(G)>=0
        done = true;
    elseif ~(P(1)>0 && Pangle>RightAngle && Pangle<LeftAngle &&
max(G)<1 && max(G)>0)
        prompt = {'Error: The Perceived sound must lie in the active
arc of the two speakers   Please enter the desired angle of the
perceived sound : '};
        answer = inputdlg(prompt);
        Pangle = str2num(answer{1});
        xP = s*cosd(Pangle);
        yP = s*sind(Pangle);
        P = [xP yP];
    end
end

figure
plot3([0 Left(1)],[0 Left(2)],[0 Left(3)],'r-o','linewidth',2)
hold on
plot3([0 Right(1)],[0 Right(2)],[0 Right(3)],'-o','linewidth',2)
plot3([0 Top(1)],[0 Top(2)],[0 Top(3)],'c-o','linewidth',2)
plot3([0 P(1)],[0 P(2)],[0 P(3)],'k-x','linewidth',2)
plot3([Left(1) Top(1)],[Left(2) Top(2)],[Left(3)
Top(3)],'k:','linewidth',2)
plot3([Right(1) Top(1)],[Right(2) Top(2)],[Right(3)
Top(3)],'k:','linewidth',2)
plot3([Left(1) Right(1)],[Left(2) Right(2)],[Left(3)
Right(3)],'k:','linewidth',2)
grid on
legend('Left Speaker','Right Speaker','Top Speaker','Virtual
Source','Active Triangle')
xlabel('feet')
ylabel('feet')
zlabel('feet')

n = sqrt(G(1)^2+G(2)^2+G(3)^2);
gL = G(1)/n;
gR = G(2)/n;
gT = G(3)/n;
fs = 44100;
perceivedL = gL.*sin(2*pi*350*(0:1/fs:2));
perceivedR = gR.*sin(2*pi*350*(0:1/fs:2));
perceivedT = gT.*sin(2*pi*350*(0:1/fs:2));
perceived = [gL.*perceivedL' gR.*perceivedR' gT.*perceivedT'];
audiowrite('perceivedL.wav',perceivedL,fs);
audiowrite('perceivedR.wav',perceivedR,fs);
audiowrite('perceivedT.wav',perceivedT,fs);
```

NonEquidistant.m

```
clear all
close all

prompt = {'Please enter the coordinates ( [x y] ) of the left speaker:
',...
    'Please enter the coordinates ( [x y] ) of the right speaker:
',...
    'Please enter the coordinates ( [x y] ) of the listener ',...
    'Please enter the desired angle of the perceived sound: '};

title = 'Speaker Parameters';
numlines = 1;
defAns = {'[10 20]','[20 20]','[12 5]','0'};
options.Resize='off';
options.WindowStyle='normal';
options.Interpreter='tex';

answer = inputdlg(prompt,title,numlines,defAns,options);
Left = str2num(answer{1});
Right = str2num(answer{2});
Listener = str2num(answer{3});
Center = [(Left(1)+Right(1))/2 Listener(2)];
dL = sqrt((Left(1)-Listener(1))^2+(Left(2)-Listener(2))^2);
dR = sqrt((Right(1)-Listener(1))^2+(Right(2)-Listener(2))^2);
s = (dL+dR)/2;
angle = str2num(answer{4})*pi/180;
x = s*sin(angle)+Listener(1);
yL = s*cos(angle)+Listener(2);
P = [x yL];
Pcenter = [P(1)+Center(1)-Listener(1) P(2)];

done = false;
while (~done)
    LeftMag = norm(Left);
    if Left(1)>0 && Left(2)>0
        done = true;
    elseif ~(Left(1)>0 && Left(2)>0)
        fprintf('Error: Speaker coordinates must be positive  \n');
        Left = input('Please input coordinates ([x y]) for the left
speaker: ');
    end
end
thetaLrad = atan((Left(1)-Listener(1))/(Left(2)-Listener(2)));
thetaLdeg = thetaLrad*180/pi
```

```
done = false;
while (~done)
    RightMag = norm(Right);
    if Right(1)>0 && Right(2)>0
        done = true;
    elseif ~(Right(1)>0 && Right(2)>0)
        fprintf('Error: Speaker coordinates must be positive \n');
        Right = input('Please input coordinates ([x y]) for the right
speaker: ');
    end
end
thetaRrad = atan((Right(1)-Listener(1))/(Right(2)-Listener(2)));
thetaRdeg = thetaRrad*180/pi

done = false;
while (~done)
    if Listener(2)<Left(2) && Listener(2)<Right(2) &&
Listener(1)>Left(1) && Listener(1)<Right(1)
        done = true;
    elseif ~(Listener(2)<Left(2) && Listener(2)<Right(2) &&
Listener(1)>Left(1) && Listener(1)<Right(1))
        fprintf('Error: Listener out of bounds, must be in front of
and between speakers \n');
        Listener = input('Please input coordinates ([x y]) for the
Listener: ');
    end
end

done = false;
while (~done)
    Pvect = (Pcenter-Center);
    thetaPrad = atan((P(1)-Listener(1))/(P(2)-Listener(2)));
    thetaPdeg = thetaPrad*180/pi
    L = [(Left-Center)' (Right-Center)'];
    G = Pvect*inv(L)
    if G(1)>1
        G(2)=G(2)/G(1)
        G(1) = 1
    elseif G(2)>1
        G(1)=G(1)/G(2)
        G(2) = 1
    end
    if dR>dL
        Gcorrect = [G(1) G(2)*(dR/dL)^2]
        TL = (dR-dL)/340.29;
        TR = 0;
    elseif dL>dR
```

```
        Gcorrect = [G(1)*(dL/dR)^2 G(2)]
        TR = (dL-dR)/340.29;
        TL = 0;
    else
        Gcorrect = G
        TR = 0;
        TL = 0;
    end
    Pcal = Gcorrect*L;

    if P(1)>0 && P(2)>0 && thetaPdeg<thetaRdeg && thetaPdeg>thetaLdeg
        done = true;
       'Please input a 2 dimensional vector ([x y]) for the desired
perceived location: ');
    end
end

zp = 5;   % padding
pixscale = 4; % scales number of pixels
ymin = min([Right(2) Left(2) Listener(2)]);
ymax = max([Right(2) Left(2) Listener(2)]);
Nrows = pixscale*(ymax-ymin+2*zp);
xmin = min([Right(1) Left(1) Listener(1)]);
xmax = max([Right(1) Left(1) Listener(1)]);
Ncols = pixscale*(xmax-xmin+2*zp);
img = zeros(Nrows,Ncols);

figure
plot([Listener(1) Left(1)],[Listener(2) Left(2)],'r-o')
hold on
plot([Listener(1) Right(1)],[Listener(2) Right(2)],'-o')
plot([Listener(1) P(1)],[Listener(2) P(2)],'k-x')
grid on
axis([0 xmax+2*zp 0 ymax+2*zp])
messageGL = num2str(Gcorrect(1));
messageTL = num2str(TL);
gtext({messageGL,messageTL})
messageGR = num2str(Gcorrect(2));
messageTR = num2str(TR);
gtext({messageGR,messageTR})
xlabel('feet')
ylabel('feet')

Npoints = 100;
lineL = inv([Left(1) 1; Listener(1) 1])*[Left(2);Listener(2)];
lineR = inv([Right(1) 1; Listener(1) 1])*[Right(2);Listener(2)];
x = linspace(xmin,xmax,Ncols);
```

```
y = linspace(ymin,ymax,Nrows);

figure
if dR>dL
    xL = linspace(Left(1),Listener(1),Npoints);
    yL = lineL(1)*xL+lineL(2);
    xR = linspace(Right(1),Listener(1),round(Npoints*dR/dL));
    yR = lineR(1)*xR+lineR(2);
    addpts = round(Npoints*dR/dL)-Npoints;
    deltaD = sqrt((xL(1)-xL(2))^2+(yL(1)-yL(2))^2);
    deltaT = deltaD/340.29;
    IL(1) = Gcorrect(1);
    IR(1) = Gcorrect(2);
    GF = 10^((log10(Gcorrect(1)/Gcorrect(2))/addpts));

    for j = 1:addpts
        [rowR(j),colR(j)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xR(j),yR(j));
        if j>1
            IR(j) = IR(j-1)*GF;
        end
        img(round(rowR(j)),round(colR(j)))=IR(j);
        logimg = log10(img+10^-1)+1;
        imagesc(x,y,fliplr(logimg))
        xlabel('feet')
        ylabel('feet')
        colorbar
        time = j*deltaT*1000;
        text(ymin+5.5,xmin-4,[num2str(time) 'ms'],'Color',[1 1 1])
        F(j)=getframe;
    end

    for i = 1:length(xL);
        j = i+addpts;
        [rowL(i),colL(i)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xL(i),yL(i));
        IR(j) = IR(j-1)*GF;
        if i>1
            IL(i) = IL(i-1)*GF;
        end
        img(round(rowL(i)),round(colL(i)))=IL(i);
        [rowR(j),colR(j)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xR(j),yR(j));
        img(round(rowR(j)),round(colR(j)))=IR(j);
        logimg = ((log10(img+10^-1)+1)/1);
        imagesc(x,y,fliplr(logimg))
        xlabel('feet')
```

```
        ylabel('feet')
        colorbar
        time = j*deltaT*1000;
        text(ymin+5.5,xmin-4,[num2str(time) 'ms'],'Color',[1 1 1])
        F(j)=getframe;
    end

elseif dL>dR
    xL = linspace(Left(1),Listener(1),Npoints*dL/dR);
    yL = lineL(1)*xL+lineL(2);
    xR = linspace(Right(1),Listener(1),round(Npoints));
    yR = lineR(1)*xR+lineR(2);
    addpts = round(Npoints*dL/dR)-Npoints;
    deltaD = sqrt((xL(1)-xL(2))^2+(yL(1)-yL(2))^2);
    deltaT = deltaD/340.29;
    IL(1) = Gcorrect(1);
    IR(1) = Gcorrect(2);
    GF = 10^((log10(Gcorrect(2)/Gcorrect(1))/addpts));

    for i = 1:addpts
        [rowL(i),colL(i)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xL(i),yL(i));
        if i>1
            IL(i) = IL(i-1)*GF;
        end
        img(round(rowL(i)),round(colL(i)))=IL(i);
        logimg = ((log10(img+10^-1)+1)/1);
        imagesc(x,y,fliplr(logimg))
        colorbar
        xlabel('feet')
        ylabel('feet')
        time = i*deltaT*1000;
        text(ymin+5.5,xmin-4,[num2str(time) 'ms'],'Color',[1 1 1])
        F(i)=getframe;
    end

    for j = 1:length(xR);
        i = j+addpts;
        [rowR(j),colR(j)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xR(j),yR(j));
        IL(i) = IL(i-1)*GF;
        if j>1
            IR(j) = IR(j-1)*GF;
        end
        img(round(rowR(j)),round(colR(j)))=IR(j);
        [rowL(i),colL(i)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xL(i),yL(i));
```

```
        img(round(rowL(i)),round(colL(i)))=IL(i);
        logimg = ((log10(img+10^-1)+1)/1);
        imagesc(x,y,fliplr(logimg))
        colorbar
        xlabel('feet')
        ylabel('feet')
        time = i*deltaT*1000;
        text(ymin+5.5,xmin-4,[num2str(time) 'ms'],'Color',[1 1 1])
        F(i)=getframe;
    end

else
    xL = linspace(Left(1),Listener(1),Npoints);
    yL = lineL(1)*xL+lineL(2);
    xR = linspace(Right(1),Listener(1),round(Npoints*dR/dL));
    yR = lineR(1)*xR+lineR(2);
    deltaD = sqrt((xL(1)-xL(2))^2+(yL(1)-yL(2))^2);
    deltaT = deltaD/340.29;
    IL(1) = Gcorrect(1);
    IR(1) = Gcorrect(2);
    addpts = round(Npoints*dL/dR)-Npoints;
    GF = 10^(log10(0.1/Gcorrect(1))/(Npoints-1));
    for i = 1:length(xR)
        [rowR(i),colR(i)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xR(i),yR(i));
        [rowL(i),colL(i)] =
xy_to_rowcol(xmin,xmax,ymin,ymax,zp,Nrows,Ncols,xL(i),yL(i));
        if i>1
            IL(i) = IL(i-1)*GF;
            IR(i) = IR(i-1)*GF;
        end
        img(round(rowL(i)),round(colL(i)))=IL(i);
        img(round(rowR(i)),round(colR(i)))=IR(i);
        imagesc(x,y,fliplr(img))
        colorbar
        xlabel('feet')
        ylabel('feet')
        time = i*deltaT*1000;
        text(ymin+5.5,xmin-4,[num2str(time) 'ms'],'Color',[1 1 1])
        F(i)=getframe;
    end
end

figure
movie(F,1,10)
```