2D AND 3D AUDIO SOUND LOCALIZATION UTILIZING VECTOR BASED AMPLITUDE PANNING

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

Kaleb Frizzell

Director: Dr. Robert Adams
Associate Professor
School of Engineering Technology

Committee Members: Dr. Peter Tay, School of Engineering Technology
Dr. Yanjun Yan, School of Engineering Technology

March 2018

ACKNOWLEDGEMENTS

The author greatly appreciates valuable assistance offered by Dr. Robert Adams, Dr. Peter Tay and Dr. Yanjun Yan. The author would like to convey his gratefulness to his friends and family for their constant support during the course of this project. I would also like to thank James Franks for helping with striping and assembling wiring.

TABLE OF CONTENTS

List of Tables.. iv
List of Figures.. v
Abstract.. vi
CHAPTER 1: Introduction.. 1
CHAPTER 2: Literature Review.. 3
2.1 2D and 3D Audio.. 3
2.2 Sound Perception.. 3
2.2.1 Sound Perception with Two Speakers.. 4
2.2.2 Sound Perception with Three or More Speakers.. 4
2.3 Vector Based Amplitude Panning.. 6
2.4 Previous Research at WCU.. 6
CHAPTER 3: Design Methodology.. 8
3.1 Four Channel 2 Dimensional System.. 8
3.1.1 2D Static Localization.. 10
3.1.2 2D Static Algorithem.. 10
3.1.3 2D Static Test Procedure.. 12
3.1.4 2D Dynamic Localization.. 14
3.1.5 2D Dynamic Algorithem.. 14
3.1.6 2D Dynamic Test Procedure.. 16
3.2 Four Channel 3 Dimensional System.. 16
3.2.1 3D Static Localization.. 19
3.2.2 3D Static Algorithem.. 20
3.2.3 3D Static Test Procedure.. 21
3.2.4 3D Dynamic Localization.. 23
3.2.5 3D Dynamic Algorithem.. 23
3.2.6 3D Dynamic Test Procedure.. 23
CHAPTER 4: Results.. 25
4.1 2D Static Sound Localization Test Results 25
4.2 2D Dynamic Sound Localization Test Results 27
4.3 3D Static Sound Localization Test Results 30
4.4 3D Dynamic Sound Localization Test Results 32
CHAPTER 5: Conclusion and Future Work.. 35
Bibliography.. 37
Appendix.. 39
APPENDIX A: Collected Data 40
APPENDIX B: MATLAB Code 50

LIST OF TABLES

Table A.1 2D static experimental data (all angles in degrees) ……………………..... 41
Table A.2 Average RSME for each calculated static angle (all angles in degrees) ... 42
Table A.3 2D dynamic experimental data participants 1 to 5 (all angles in degrees) . 43
Table A.4 2D dynamic experimental data participants 6 to 10 (all angles in degrees) 44
Table A.5 Average RSME for each calculated dynamic start and dynamic end angle (all angles in degrees) ……………………………………..……. 45
Table A.6 3D random angle test position (all angles in degrees) …………….…... 46
Table A.7 3D static localization experimental data …………………….……... 47
Table A.8 Average RSME for each 3D static location ……………………………… 48
Table A.9 3D random angle test positions for the start and end angles ……………... 48
Table A.10 3D dynamic localization experimental data ……………………………... 49

LIST OF FIGURES

Figure 3.1 Two channel vector based amplitude Panning …………………………….... 9
Figure 3.2 Four channel 2D static simulation example …………………..........…….. 11
Figure 3.3 Four channel 2D experimental test setup ………………………………… 12
Figure 3.4 Block diagram of audio speaker ……………………………………….. 13
Figure 3.5 Four channel 2D dynamic simulation example …………………………... 15
Figure 3.6 Three channel configuration for 3D sound localization ………………….. 17
Figure 3.7 Virtual source locations ……………………………………………….. 20
Figure 3.8 Four channel 3D experimental test setup ………………………………… 22
Figure 3.9 Four channel 3D dynamic arc paths ……………………………………… 24
Figure 4.1 Four channel 2D static RSME ……………………………………...…. 26
Figure 4.2 Four channel 2D dynamic start angle RSME …………………...……….. 28
Figure 4.3 Four channel 2D dynamic end angle ……………………………….…..... 29
Figure 4.4 Four channel 3D static localization - percent of correct responses ……… 30
Figure 4.5 Four channel 3D static localization - percent of responses within 15 degree …..………………………………………………………………. 31
Figure 4.6 RSME for each virtual source location …………………………………. 32
Figure 4.7 Four channel 3D static localization - percent of correct ………………… 33
Figure 4.8 Four channel 3D static localization - percent of responses within one path 34

ABSTRACT

2D AND 3D AUDIO SOUND LOCALIZATION UTILIZING VECTOR BASED AMPLITUDE PANNING

Kaleb Frizzell, M.S.T.

Western Carolina University (March 2018)

Director: Dr. Robert Adams

Audio systems are used to create two-dimensional (2D) and three-dimensional (3D) audio effects which involve the ability to localize sound within a multi-dimensional space. Multi-dimensional audio systems could be used to imitate moving sounds in applications such as home theaters, video games or headphones. When two or more equidistant speakers produce the same sound, the observer will perceive the sound to be localized at a single point. The blending of sound from equidistant speakers is called the virtual sound and is perceived to originate from a virtual source. For two speakers, the virtual source is located on a circular arc between the speakers and for three speakers, the virtual source is located on a spherical cone defined by the speakers. For the observer to perceive one sound from multiple sources, the sounds must arrive at the observer at the same time and the sounds must be the same. By calculating the individual speaker gains using the method of vector-based amplitude panning (VBAP), the audio from all the speakers can be manipulated such that the observer perceives the sound to be originating from a single point. The objective of this project was to develop an algorithm that can place an audio tone in the desired location by calculating and controlling the gain factors of each speaker. In this thesis, the results of simulating in MATLAB and testing in the lab, two-dimensional (2D) and three-dimensional (3D) audio systems with multiple speakers placed in testing positions equidistant to the observer are presented. It is envisioned that this research will lead to a better understanding of localization of sound and to a better understanding of how accurately sound is perceived by the human ear

CHAPTER 1: INTRODUCTION

Humans naturally hear in 3D, meaning they can perceive and localize the source of sounds. The localization of a perceived sound source can be identified by the following three coordinates: azimuth, elevation, and distance. Azimuth is the horizontal angle from the medium plane; elevation is the angle between the horizontal plane and the perceived sound and distance is how far the source of the sound is away from the observer. To determine the azimuth, our ears use the slight differences in time and pressure of a sound as it reaches the left and right ears to locate the source of the sound without seeing it [1]. The cues used to determine the azimuth of a sound don't 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. 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.

Using software to generate audio and a specific speaker test set up, we can control these parameters and have the human ears perceive a sound in the desired place. Over the past few years, there have been advances in audio technology in both stereo systems, and headphones and these systems provide a more realistic listening experience. Even though there have been many advances, there is still a way to go in creating a fully realized 3D audio experience. Edward Matthews [2] researched this topic and I plan to extend his findings and explore more aspects of 2D and 3D audio technology.

In this research, a MATLAB code was developed that generated all the necessary audio files by applying the calculated gain factor to audio files, and after getting IRB approval, the researcher tested the audio set up with the generated audio files to determine how well humans perceive sound. In this thesis, I present the various papers I read and researched, the design and methodology of the experiment and the results were obtained from the experiments. The results will be analyzed and explained, followed by an explanation to how the experiment could be possibly expanded in the future.

CHAPTER 2: LITERATURE REVIEW

2.1 2D and 3D Audio

Over the past century electronics have progressed dramatically, and even though sound quality has improved over the years, stereo systems have not seen the same amount of progression. Current stereo setups cannot imitate real life perfectly meaning that sounds coming from the speakers seem always to be originating from the same origin. 2D and 3D audio would fix this current issue by allowing sounds to seem to appear from anywhere in the room which would give the user a more realistic listening experience when using any device that produces sound. To achieve this, the sound wave's gain values are manipulated to make the sound appear where you want them to be.

2.2 Sound Perception

The study of the human perception of sound is called psychoacoustics, and sound localization is the process of determining the location of a sound source. When the human ear receives sound waves, the brain utilizes subtle differences in loudness, tone, and timing between the two ears to allow humans to localize sound [1]. Localization can be described regarding its three-dimensional position. The azimuth (horizontal angle), the zenith (vertical angle), and the distance (for static sounds) [3]. Humans are adept at detecting the direction in the horizontal plane due to their ears being placed symmetrically on the head. The ears being symmetric makes it harder to perceive the vertical plane.

The factors that go into a sound are its intensity, frequency, and overtones. Frequency is perceived as a pitch, and the sound intensity of the sound wave is what humans can hear. The range humans can hear ranges from 20 Hz to 20,000 Hz [4]. Our ears use interaural cues such as slight differences in time and pressure of a sound as it reaches the left and right ear to localize sound [5]. It allows them to detect where the source is by allowing them to detect which direction it is coming from. In the case where the source is directly in front of a person, the distance that the sound waves travel to each ear is going to be equal. Therefore, there will be no interaural differences to distinguish where the sound is coming from. In this specific case, the ears use the head-related transfer function (HRTF) which is used to determine the elevation [6]. As a sound wave travels through the air to the eardrums, the signal is filtered through the head and torso. The differences in the intensities of the frequencies compared to the eardrum and torso make up the HRTF [6].

2.2.1 Sound Perception with Two Speakers

The precedence effect in sound localization states that when two sound waves reach the ear in a specific amount of time, then the sound will be perceived as a single auditory event [7]. The most common multi-dimensional audio system is the two-channel stereophonic configuration. Stereophonic sound is a method of sound reproduction that creates an illusion of multidimensional audible perspective. This is usually achieved by using two independent audio channels through a configuration of two loudspeakers (or stereo headphones) in such a way as to create the impression of sound being heard from various directions, as in natural hearing.

2.2.2 Sound Perception with Three or More Speakers

Surround sound is a technique for enriching the sound reproduction quality of an audio source with additional audio channels from speakers that surround the listener providing sound from within a 360° circle in the horizontal plane. The technique enhances the perception of sound localization. This is typically achieved by using multiple audio channels routed to an array of loudspeakers [8]. An example of a surround sound system is Dolby 5.1 which generates sounds from different directions, and the source of the sound seems to be localized at the speaker that it was generated from. Surround sound is not able to make the sound appear to be coming from above or below the speakers due to it being localized at the source.

Headphones are an approach that can be utilized to generate 3D audio. Wightman performed experiments comparing the localization of sound presented in a free field environment to headphones [9]. According to the results of the experiment, headphones could imitate a free field environment in the horizontal plane but could not perfectly recreate the free field environment entirely. Even though headphones can recreate 3D audio, it has the drawback of being limited to a single listener.

All the pre-mentioned techniques that have been presented share a common disadvantage of being confined to one listener. They all only work well when the user is at a specific location, or when they are confined to a small listening space. Therefore, to achieve 3D audio that can be listened by a larger audience, a multi-channel should be implemented. By adding more speakers, there are more paths on which the virtual source can move [10].

Pulkki presents a simple model in [10] where a single elevated speaker is introduced to the stereo configuration. The elevated speaker described is the same distance as the two speakers on the horizontal plane. The three speakers will form a section of a 3-D sphere that they refer to as the active triangle on which the virtual source can be positioned anywhere within the active triangle.

In Pulkki's work, they describe how it is easy to say that when the number of speakers that are in a system increases the more accurate it will be, however as the number of speakers increases, so does the cost and amount of space required for the system. Therefore, for this research effort, the number of channels was limited to four speakers.

2.3 Vector Based Amplitude Panning

In an article, Ville Pulkki describes how to use vector based math using the sound vectors to calculate the necessary gains for the standard two-channel setup [10]. When the same signal is transmitted on the pre-mentioned channels, the sound will be perceived as a single auditory event; this event is known as a virtual sound source, coming from somewhere between the speakers and the position between the two speakers is determined based on the gain factors [11]. That position will lie on what Pulkki calls the active arc, where the radius of the arc will be determined by the distance of the speakers. The virtual source can be positioned anywhere between the two speakers but lacks the ability to move anywhere outside of this arc [12, 13]. Because of this flaw, many techniques have been experimented with to move the perceived source outside of this boundary [14] and to increase the size of the area where the listener can localize the sound [11]. Research has also been done on how to make the sound move with the listener [15, 16].

Gain refers to the amplification factor and is the extent to which an analog amplifier boosts the strength of a signal. Adjusting the gain of each channel is known as intensity panning [17]. In the situation where the two loudspeakers are positioned symmetrically to the median plane, their gains will be equal.

2.4 Previous Research at WCU

Dr. Adams' previous graduate student, Edward Matthews, performed research on this topic in 2016. Matthews worked with three speakers to study 2D and 3D audio. During his research, he tested 2D and 3D test setups where all the listeners were all the same distance apart from speakers. He found out that people could accurately localize sound in the horizontal plane, but they had difficulty localizing sounds in the vertical plane. He also simulated, in MATLAB, a situation where two speakers were not the same distance apart from an observer [2]. It is envisioned that this research will expand our knowledge of how the human ears perceives sounds which was achieved by expanding his 2D and 3D test setups by testing new configurations of speakers for both 2D and 3D audio systems.

CHAPTER 3: DESIGN METHODOLOGY

3.1 Four Channel 2-Dimensional System

This research began by reviewing the current literature on the use of sound localization, specifically in the areas of 2D and 3D audio. Some preliminary simulations were undertaken to understand the process of generating the correct audio files. The literature review and theoretical study were followed by planning the experimentation required for testing 2D and 3D audio systems. This included acquiring parts such as speakers, audio amplifiers, digital audio interface, sound forge and any other equipment required for testing.

In this section our investigation of the vector based amplitude panning (VBAP) discussed in Pulkki's paper [10] is 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 distant away 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 sections will discuss the mathematical derivation, algorithm development, and testing procedure for four channel 2D and 3D localization, which both implement amplitude panning.
Two-dimensional vector based amplitude panning is the base used in the setup that was implemented. At any given time only two of the four channels will be active to create a 2D sound environment. Figure 3.1 is a visual representation of a basic two-dimensional setup and its sound vectors.

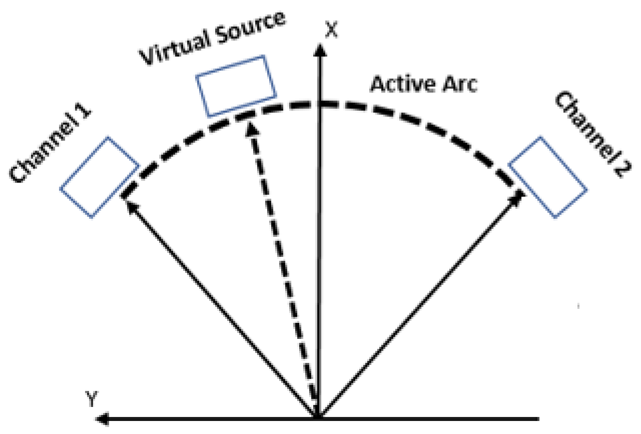

Figure 3.1: Two channel vector based amplitude panning

In Figure 3.1, Channel 1 ($\boldsymbol{l_1}$) and Channel 2 ($\boldsymbol{l_2}$) are the position vectors for the left and right speakers, and the virtual source ($\boldsymbol{p}$) is the virtual source vector. The speaker vectors can be expressed using Equation (3.1) where *m* is the channel number, l_{mx} is the x component of the vector and l_{my} is the y component of the vector.

$$\boldsymbol{l_m} = [l_{mx} \quad l_{my}] \tag{1}$$

The x and y position of the speakers are determined using

$$l_{mx} = s\cos(\theta_m) \tag{2}$$

$$l_{my} = s\sin(\theta_m) \tag{3}$$

Where *m* is the channel number, *s* is the distance from the observer to the speakers, θ_m are the azimuth angles from the x axis for each speaker.

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 non-negative 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:

$$C = {g_1}^2 + {g_2}^2 \tag{4}$$

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 toward the virtual source can be represented as a linear combination of the speaker vectors and their respective gains.

$$\boldsymbol{p} = [p_x \quad p_y] = l_1 g_1 + l_2 g_2 \tag{5}$$

Equations (3.6) and (3.7) are utilized to determine the x and y position of the virtual source, where *m* is the channel number, *s* is the distance from the observer to the speakers and θ_p are the angles from the x axis for the virtual source.

$$p_x = s\cos(\theta_p) \tag{6}$$

$$p_y = s\sin(\theta_p) \tag{7}$$

3.1.1 2D Static Localization

The first test that was conducted was static localization in which the virtual source was specified to be at a stationary angle (making the resulting speaker gains constant).

3.1.2 2D Static Algorithm

MATLAB code was developed that calculates the necessary gain factors required to position the virtual source at a given location. The gain, which refers to the amplification factor, is represented by the variable $\boldsymbol{g}$ and is the extent to which an analog amplifier boosts the strength of a signal. To calculate the necessary gain factors, the following formula was used.

$$\boldsymbol{g} = p^t {l_{123}}^{-1} = [p_x \quad p_y \quad p_z] \begin{bmatrix} l_{1x} & l_{1y} & l_{1z} \\ l_{2x} & l_{2y} & l_{2z} \\ l_{3x} & l_{3y} & l_{3z} \end{bmatrix}^{-1} \tag{8}$$

When the MATLAB code is run, the user is asked to input the location of the speakers by defining the angle from the x-axis to the left and right speakers and to define the angle for the desired virtual source. The MATLAB code generated two 2 second, 400 Hz audio signals with a sampling frequency of 44.1 kHz. The individual speaker gains calculated from (3.8) were applied to each signal to produce the tones needed for each channel.

Before testing on participants, simulations for 2D VBAP were tested and performed in MATLAB. Figure 3.2 is a simulation of static testing in which the left and right speakers are both 6 feet from the observer.

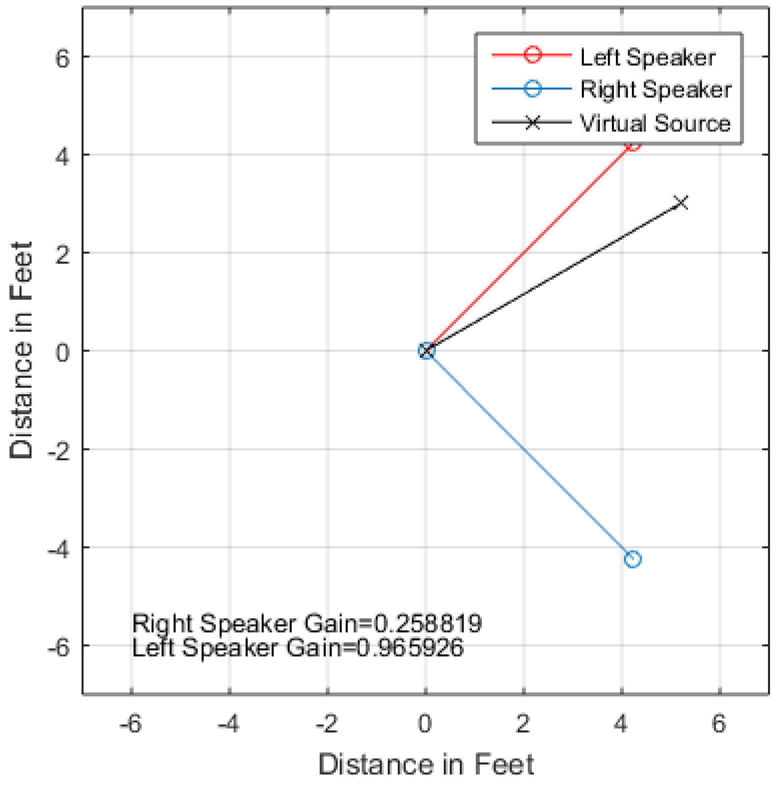

Figure 3.2: Four channel 2D static simulation example

The right speaker is 45° above the x-axis, and the left speaker is 45° below the x-axis. The user specifies a virtual source located on the active arc, 30° above the x-axis. The VBAP algorithm calculated left and right speaker gains of 0.9659 and 0.2588, respectively, to generate the specified virtual source.

3.1.3 2D Static Test Procedure

A specific test setup was implemented that involved four channels which were used to create 2D audio effects. It is important to note that at any given time, only two of the four channels will be active at any given time. The setup for 2D audio that was implemented had four speakers that were all positioned to be six feet away from the observer. Two of the speakers were in front of the observer, and the other two speakers were located behind the observer. The observer sat in the exact center of the speakers, and the speakers were placed at azimuth angles 45°, 135°, 225° and 315° relative to the observer. All four speakers were on the same XY plane. The observer sat in the exact center of the circle of speakers with the head faced an azimuth of 90°. This setup does not simulate 3D audio. Since all the speakers are in one plane surrounding the observer, it is an implementation of 360° 2D audio. This setup was chosen so we could test to see if humans have difficulty localizing sound behind them when compared to sounds in front of them. The following figure is a photograph of the test setup that was implemented.

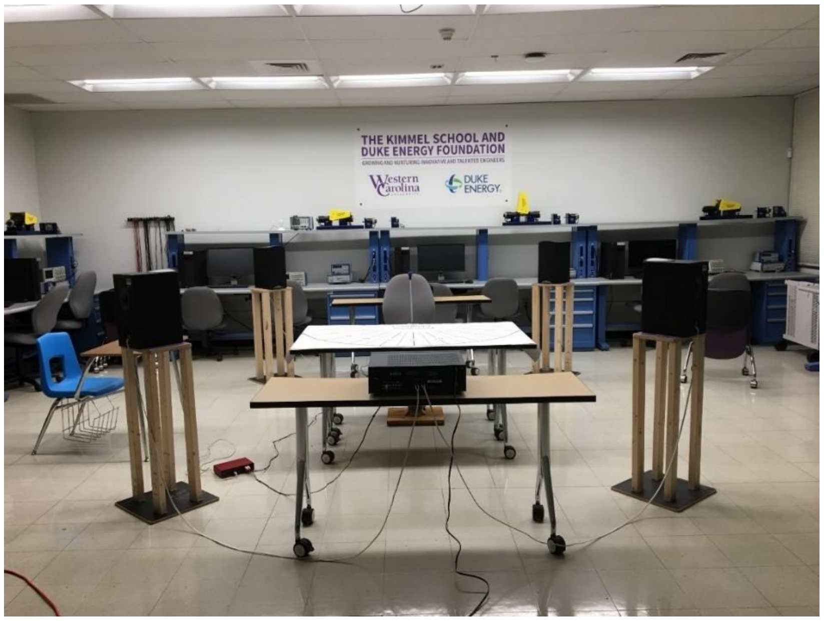

Figure 3.3: Four channel 2D experimental test setup

MATLAB was used to create Waveform Audio Files (.wav) files which are the audio files that were used in the testing phase of our research. A .wav file is an audio file format standard for storing an audio bit stream onto PCs. MATLAB cannot 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. Once the .wav files were generated, they were imported into Sound Forge, a digital audio editing application, and assigned to different channels. A base signal was generated to be used in participant testing. This signal was a 400 Hz sine wave with a sampling frequency of 44.1 kHz and a duration of 2 seconds.

A digital audio workstation (DAW), SoundForge Pro 11, took those .wav files and assigned them to four independent channels. The laptop exported the four independent channels to the Digital Audio Interface (Focusrite Scarlett 2i4) via a USB cord. The DAI was connected to two audio amplifiers (Sherwood RX-4109) which controlled the gains of each of the individual (Klipsch B-3 bookshelf) speakers. Each stereo audio amplifier controlled the gains of two of the four speakers. Figure 3.4 is a block diagram that visually represents how each component is connected.

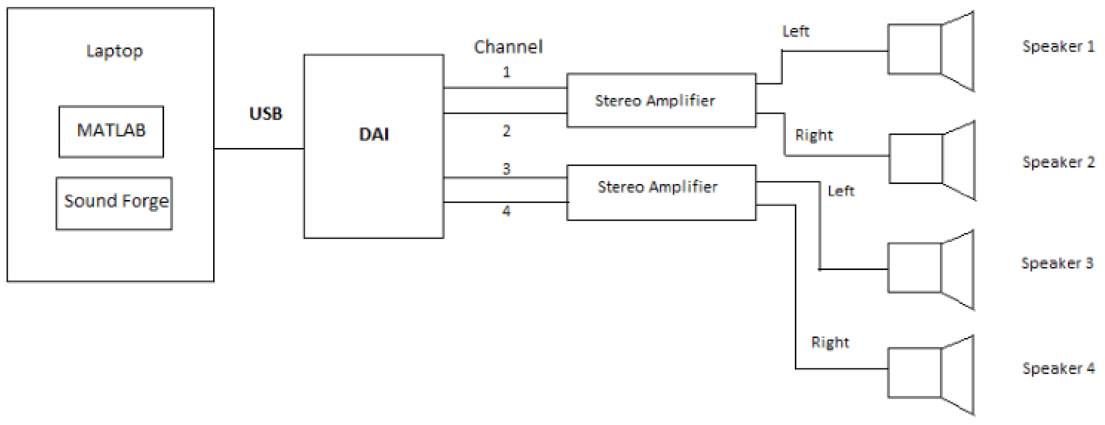

Figure 3.4: Block diagram of audio speaker setup

During testing the observer was placed in a chair that was surrounded by a set of audio speakers. The researcher explained how the testing would be conducted and gave a demonstration of the types of sounds that were played. The researcher played the base signal localized at 24 different fixed locations around the observer. The locations tested ranged from 0° to 345° with a step size of 15°. The researcher played the sounds in random order. The observer identified the angular position of each sound by pointing to the perceived angle on one of two large paper printouts of a protractor. One protractor was laid in front of the observer, and the other was positioned behind the observer.

3.1.4 2D Dynamic Localization

The second test involved dynamic testing in which the virtual source is specified to move across an active arc. The resulting speaker gains would therefore dynamically change over time. The researcher played the 2 second 400 Hz base tone targeted to a specific active arc. The observer would then indicate on the printed protractors where they thought the tone started and where the tone ended. This process was repeated for a set of 24 different active arcs. Each arc had a width of 90° and an arbitrary starting angle.

3.1.5 2D Dynamic Algorithm

In this algorithm, the gains are dynamically changing over time to create the sensation of a moving sound. To generate these audio files, 100 gain factors were calculated in MATLAB at evenly spaced time intervals and then those values were interpolated and applied to a sound wave.

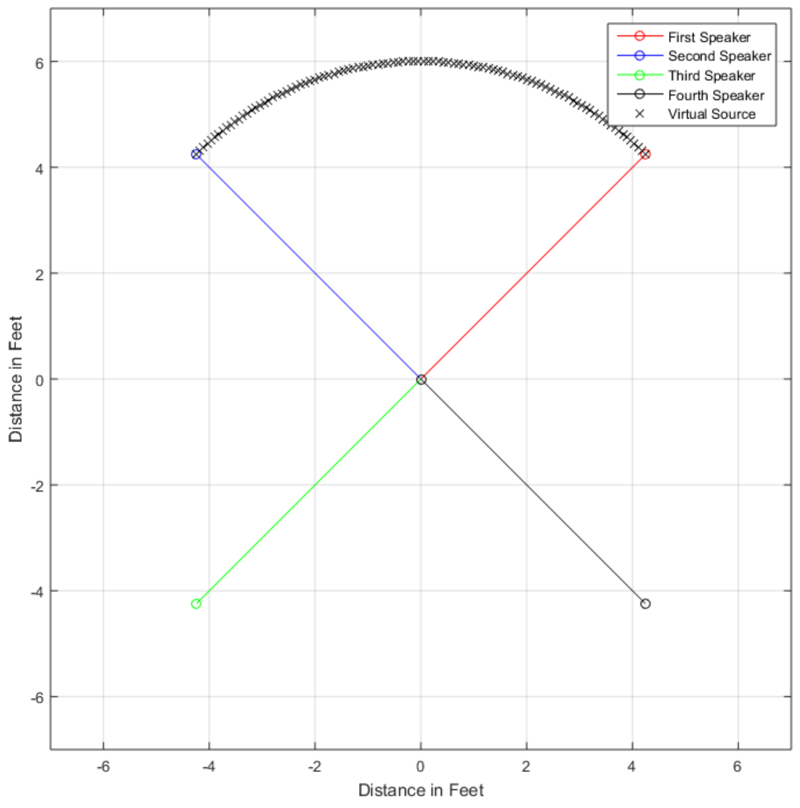

Figure 3.5: Four channel 2D dynamic simulation example

Figure 3.5 is a simulation of dynamic testing in which the all the speakers are 6 feet from the observer. The right speaker (First Speaker) is 45° to the right of the y-axis and the left speaker (Second Speaker) is 45° to the left of the y-axis. The user specifies the starting angular position of the virtual source, the total angle of the arc, and whether the sound rotates clockwise or counter-clockwise. In this case, the virtual source was specified to move from the first speaker to the second speaker. The VBAP algorithm calculated the gain factors at 100 evenly spaced intervals between the first and second speaker. The 100 gain factors were interpolated and applied to the tone files to achieve dynamic movement of sound.

3.1.6 2D Dynamic Test Procedure

The second test involved 2D dynamic testing in which the virtual source is specified to move across an active arc. The test setup used in static localization was utilized here. The resulting speaker gains dynamically changed over time. The researcher played the 2 second 400 Hz base tone targeted to a specific active arc. The researcher played the sound signals localized at 24 different arcs around the observer. The starting arc locations tested ranged from 0° to 345° with a step size of 15°. The researcher played the sound signals in a random order. Each sound was targeted to move through a 90° arc in either a clockwise or counterclockwise direction. The observer would then indicate on the printed protractors where they thought the tone started and where the tone ended. This process was repeated for the set of 24 different active arcs.

3.2 Four Channel 3-Dimensional System

Three-dimensional vector based amplitude panning is the basis for the algorithm used to generate 3D sound localization. Figure 3.6 is a visual representation of three-dimensional sound vectors. The three speakers form an active triangle in which one may position the virtual source.

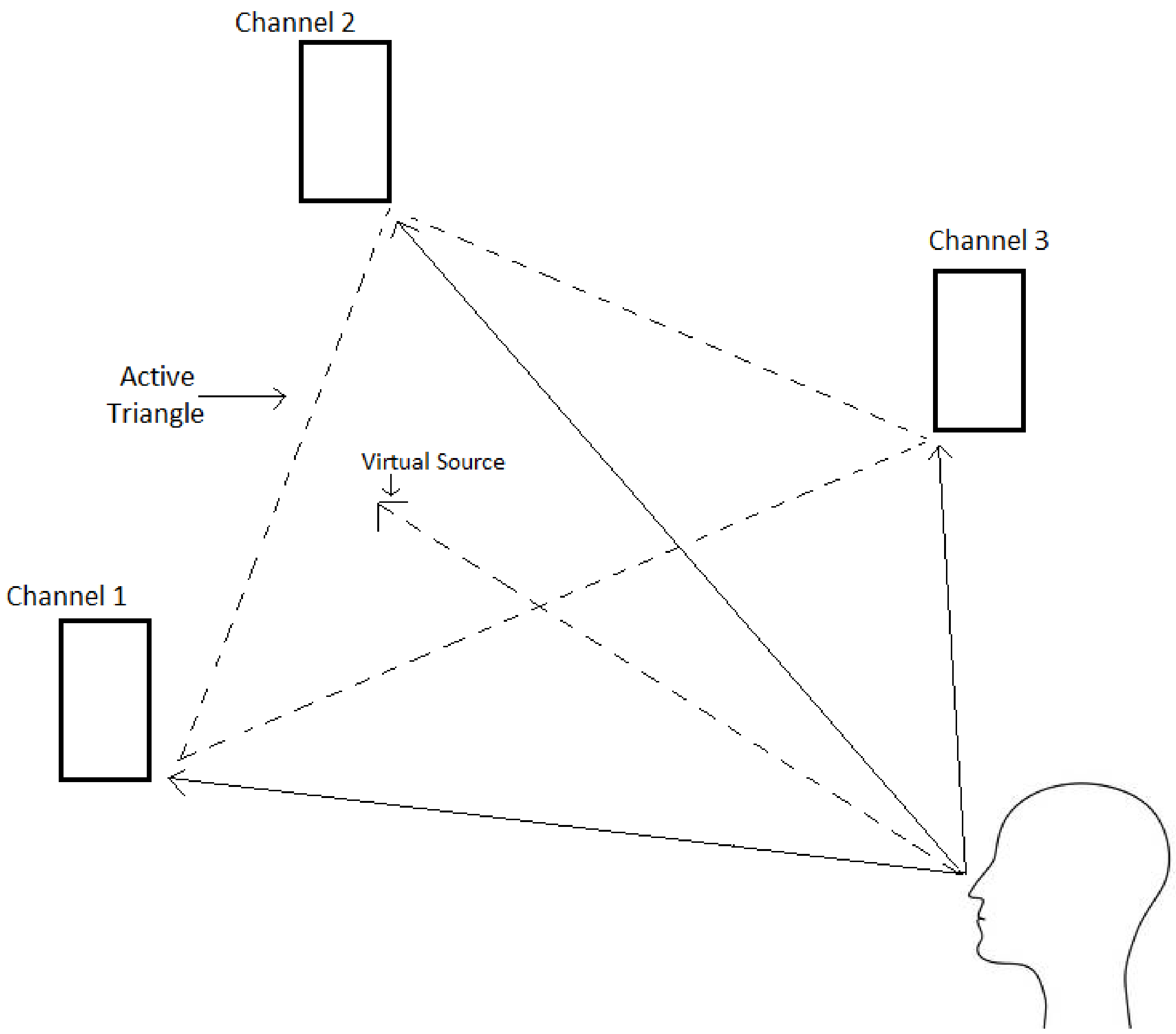

Figure 3.6: Three channel configuration for 3D sound localization

In Figure 3.6, Channel 1 ($\boldsymbol{l_1}$), Channel 2 ($\boldsymbol{l_2}$) and Channel 3 ($\boldsymbol{l_3}$) are the position vectors for the left, right and elevated speakers, and $\boldsymbol{p}$ is the virtual source vector.

The speaker vectors can be expressed using Equation 3.9 where *m* is the channel number, l_{mx}, l_{my} and l_{mz} are the x, y, z components, respectively of the position vectors.

$$\boldsymbol{l_m} = [l_{mx} \quad l_{my} \quad l_{mz}] \tag{9}$$

The x, y and z position of the speakers may be calculated using:

$$l_{mx} = s\cos(\theta_m) \tag{10}$$

$$l_{my} = s\sin(\theta_m) \tag{11}$$

$$l_{mz} = s\sin(\phi_m) \tag{12}$$

where *m* is the channel number, *s* is the distance from the observer to the speakers, θ_m is the azimuth angle from the x axis for each speaker and ϕ_m) is the elevation angle for each speaker. Note that the elevation angle for the right and left speakers is zero for the configuration shown in Figure 3.6.

The gain factors of the left, right and elevated speakers, g_1, g_2 and g_3, are non-negative 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:

$$C = {g_1}^2 + {g_2}^2 + {g_3}^2 \tag{13}$$

where *C* is the volume of the virtual source. As *C* increases, the virtual source is perceived to move closer to the observer. If the largest gain factor is greater than 1, then each gain is divided by a calculated 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

$$n = \sqrt{{g_1}^2 + {g_2}^2 + {g_3}^2} \tag{14}$$

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.

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

$$\boldsymbol{p} = [p_x \quad p_y \quad p_z] = \boldsymbol{l_1} g_1 + \boldsymbol{l_2} g_2 + \boldsymbol{l_3} g_3 \tag{15}$$

Equations (3.16) through (3.18) are utilized to determine the x, y and z coordinates for the virtual source where, *m* is the channel number, *s* is the distance to the speakers, θ_p is the azimuth angle of the virtual source, and θ_p is the elevation angle for the virtual source.

$$p_x = s\cos(\theta_p) \tag{16}$$

$$p_y = s\sin(\theta_p) \tag{17}$$

$$p_z = s\sin(\phi_p) \tag{18}$$

3.2.1 3D Static Localization

The 3D testing involved two parts, static and dynamic localization. In 3D static localization, the virtual source was specified to be at a stationary point (making the resulting speaker gains constant). The researcher played the base signal localized at 17 different fixed locations in front of the observer. The participant had a sheet that had a plot of the potential positions that the virtual source could be coming from. Figure 3.7 shows all 17 locations that were tested.

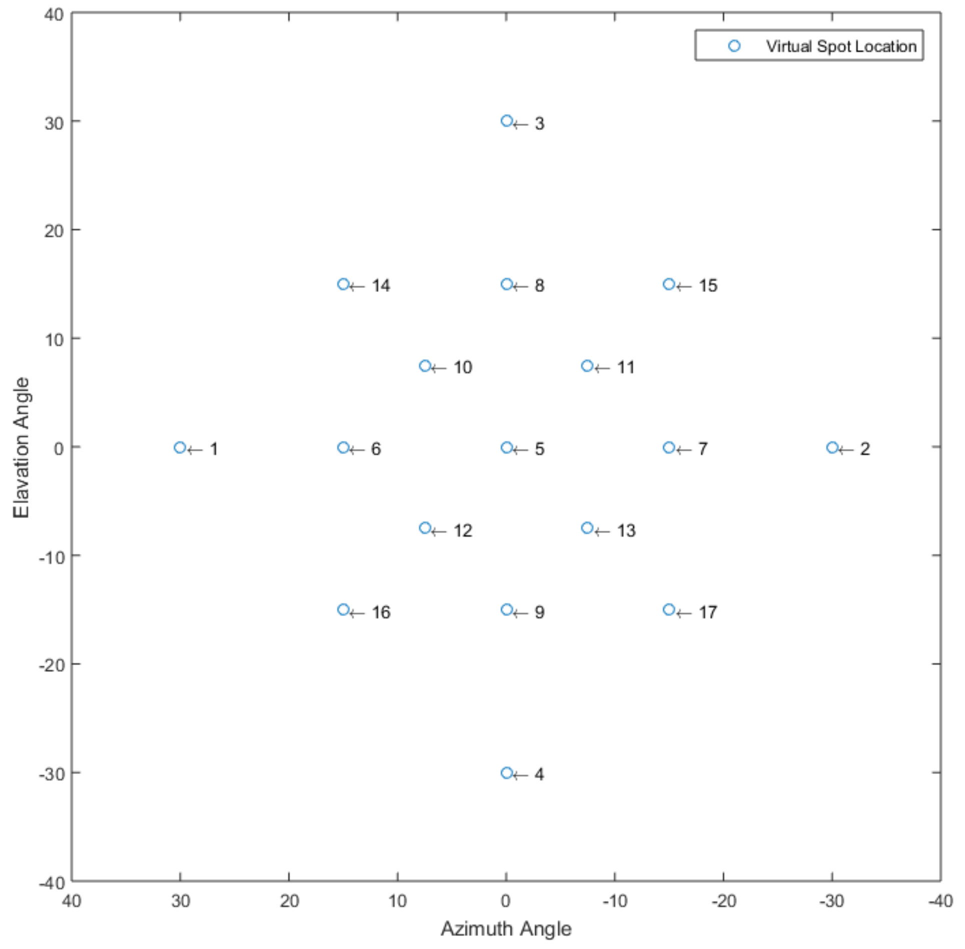

Figure 3.7: Virtual source locations

The sounds were played from one to seventeen 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.

3.2.2 3D Static Algorithm

To calculate the gains for 3D testing the following equations were used. Equation 3.19 calculates the three gain factors for the configuration shown in Figure 3.6, given the coordinate positions of the 3 speakers and the virtual source. When the MATLAB code is run, the user is asked to input the azimuth and elevation angles of each speaker and of the virtual source. The

MATLAB code then calculates the coordinates of the speaker locations using equations (3.10) through (3.12) and the coordinates of the virtual source location using equations (3.16) through (3.18).

$$\boldsymbol{g} = \boldsymbol{p}^T \boldsymbol{l_{123}}^{-1} = [p_x \quad p_y \quad p_z] \begin{bmatrix} l_{1x} & l_{1y} & l_{1z} \\ l_{2x} & l_{2y} & l_{2z} \\ l_{3x} & l_{3y} & l_{3z} \end{bmatrix}^{-1} \tag{19}$$

MATLAB applies these calculated gain factors to three of the four audio channels, because only three speakers are needed to create the active triangle shown in Figure 3.6. The sound is then outputted to the four speakers via the Focusrite Scarlett 2i4 and audio amplifiers.

3.2.3 3D Static Test Procedure

During testing the observer was placed in a chair that faced four speakers. The researcher explained how the testing would be conducted and gave a demonstration of the types of sounds that would be played. A test setup was implemented using four channels which were used to create 3D audio effects. It is important to note that at any given time, only three of the four channels were active at any given time. The left speaker was placed at 30° azimuth and 0° elevation. The right speaker was placed at -30° azimuth and 0° elevation. The top speaker was placed at 0° azimuth and 30° elevation. The bottom speaker was placed at 0° azimuth and -30° elevation. All four speakers were placed 6 feet from the listener. This setup was chosen so that we could test to see if humans have difficulty localizing sounds with varying elevations and azimuth. A photo of the 3D test setup in shown in Figure 3.8.

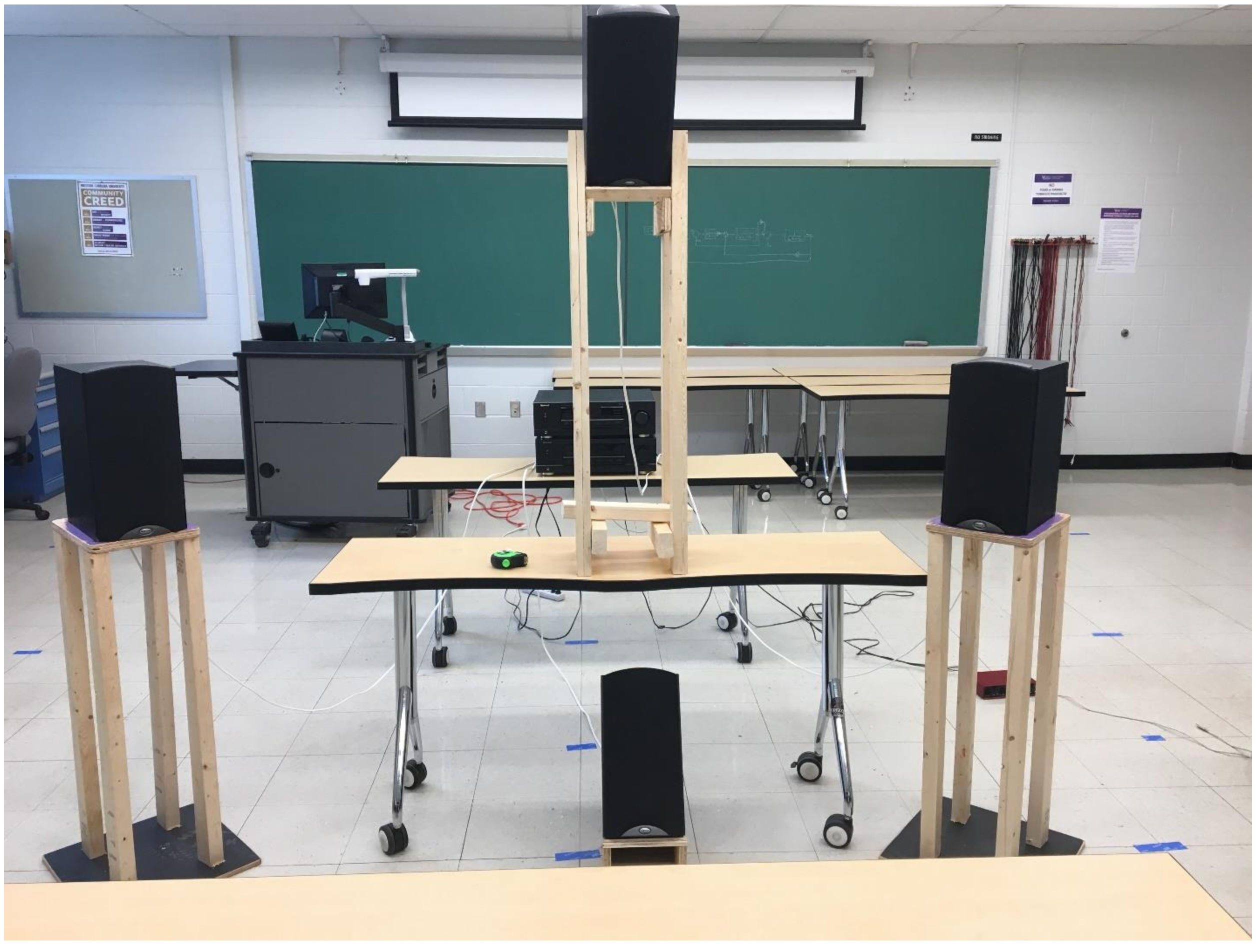

Figure 3.8: Four channel 3D experimental test setup

MATLAB was used to create Waveform Audio Files (.wav) files which are the audio files that were used in the testing phase of our research. A .wav file is an audio file format standard for storing an audio bit stream onto PCs. MATLAB cannot 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. Once the .wav files were generated, they were imported into Sound Forge, a digital audio editing application, and assigned to different channels. A base signal was generated to be used in participant testing. This signal was a 400 Hz sine wave with a sampling frequency of 44.1 kHz and a duration of 2 seconds.

A digital audio workstation (DAW), SoundForge Pro 11, took those .wav files and assigned them to four independent channels. The laptop exported the four independent channels to the Digital Audio Interface (Focusrite Scarlett 2i4) via a USB cord. The DAI was connected to two audio amplifiers (Sherwood RX-4109) which controlled the gains of each of the individual (Klipsch B-3 bookshelf) speakers. Each stereo audio amplifier controlled the gains of two of the four speakers.

3.2.4 3D Dynamic Localization

The second test involved dynamic testing in which the virtual source is specified to move across an active arc. The resulting speaker gains would therefore dynamically change over time. The researcher played the 2 second 400 Hz base tone targeted to a specific active arc. This process was repeated for a set of eight different active arcs.

3.2.5 3D Dynamic Algorithm

In this algorithm the gains are changing over time to create the sensation of a moving sound. To generate these audio files, 100 gain factors were calculated in MATLAB corresponding to evenly spaced azimuth angles across the targeted active arc at 100 evenly spaced time intervals within the 2 second sound duration. These gain factors were interpolated to the 44.1 kHz sampling frequency, and the interpolated gain factors were then applied to the 2 second 400 Hz base tone.

3.2.6 3D Dynamic Test Procedure

The second test involved dynamic testing in which the virtual source is specified to move across an active arc. The test setup is the same as the one used in 3D static localization. The researcher played the 2 second 400 Hz base tone targeted to a specific active arc path. The paths tested can be seen in Figure 3.9.

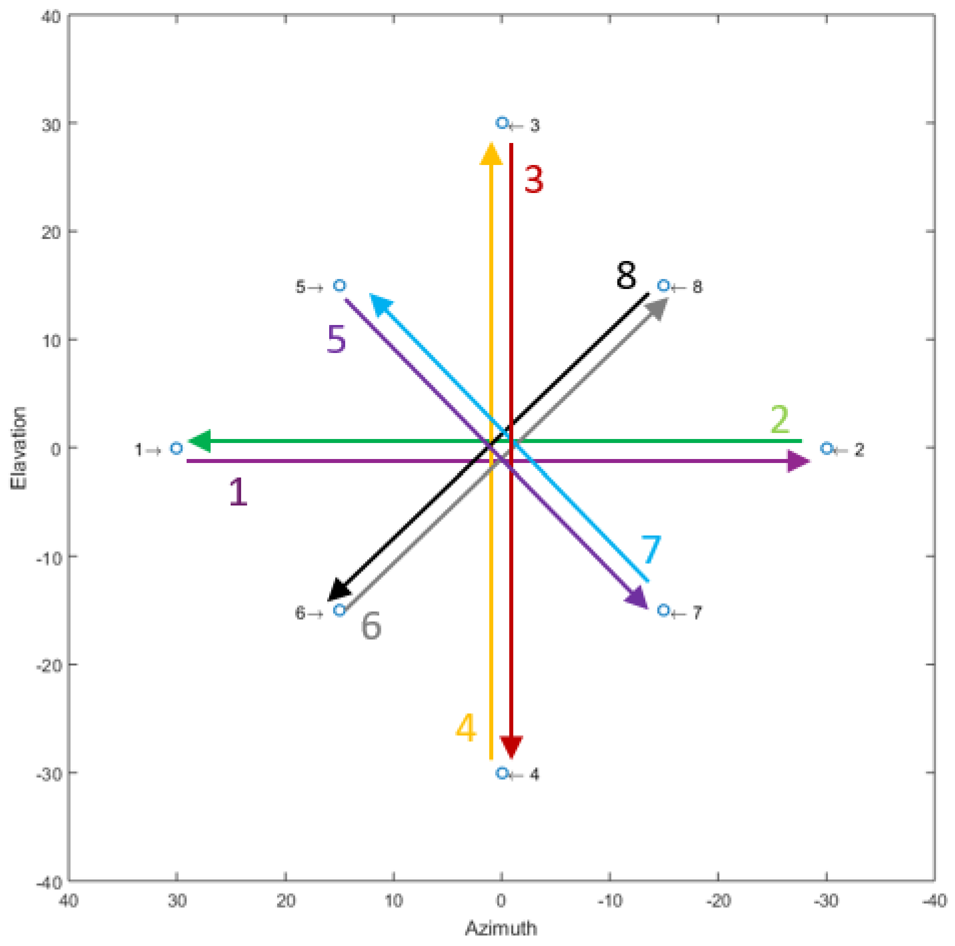

Figure 3.9: Four channel 3D dynamic arc paths

The observer was given a list of possible paths that the sound could have moved on and they would identify which path they thought the sound was traveling on. This process was repeated 8 times, each for one of the 8 designed paths.

CHAPTER 4: RESULTS

In this chapter the results of the four-channel localization testing for both static and dynamic testing, are presented. For the static and dynamic channel localization tests, experimental data was collected from participants and put into tables. This data can be found in the Appendix Section A.1 through A.9.

4.1 2D Static Sound Localization Test Results

A total of 10 participants were tested to identify the perceived azimuth angle of static tones, and the responses were compared to the calculated azimuth angle of the virtual source by calculating the Root Mean Squared Error (RSME). The RSME was calculated using

$$RSME = \sqrt{\frac{\sum_{i=1}^{n} {Error_i}^2}{n}}$$

where $Error_i$ is the difference between the perceived and actual angle, and n is the number of participants. On average the RMSE was 37.35°.

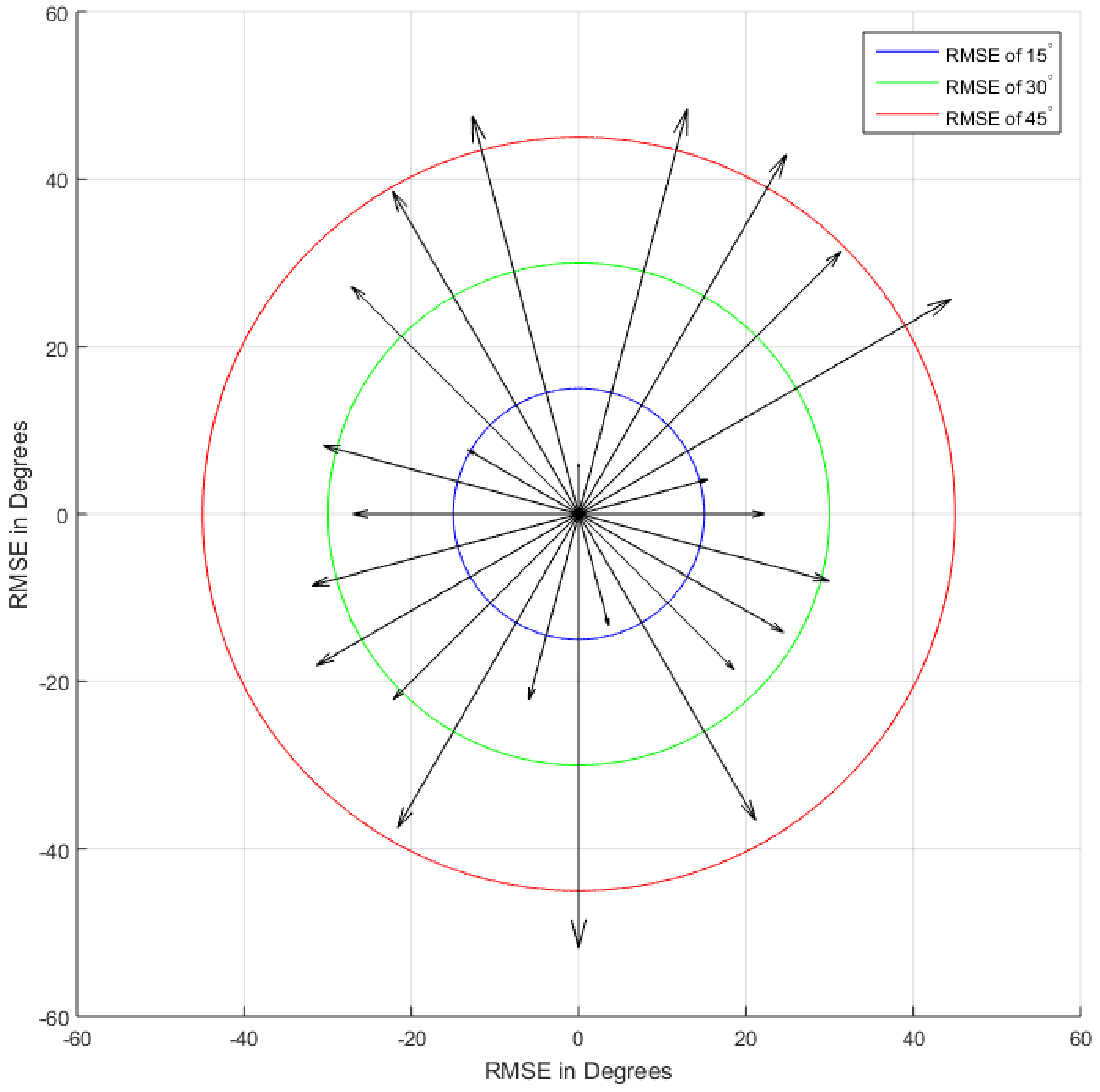

Figure 4.1: Four channel 2D static RSME

Figure 4.1 is a representation of the RMSE for each calculated angle. In this figure, each vector represents each calculated angle and the length of the vector corresponds to the RMSE value. The results indicate that the participants do not have trouble localizing sound when the tone is right in front of them. The areas that they have trouble localizing sound were when the tone was calculated to be diagonally in front or behind them and when the tone is directly behind of them.

4.2 2D Dynamic Sound Localization Test Results

A total of 10 participants were tested for dynamic localization. The results from all the participants were compared to the calculated location by calculating the RSME for the intended start angle and the intended end angle. The average RSME for all participants in dynamic testing was equal to 50.21° for the calculated start angle. On average the RMSE for the end angle was equal to 47.74°.

Figure 4.2 is a representation of the RMS error (for all 10 participants combined) between the targeted start angle and the perceived start angle as a function of targeted start angle.

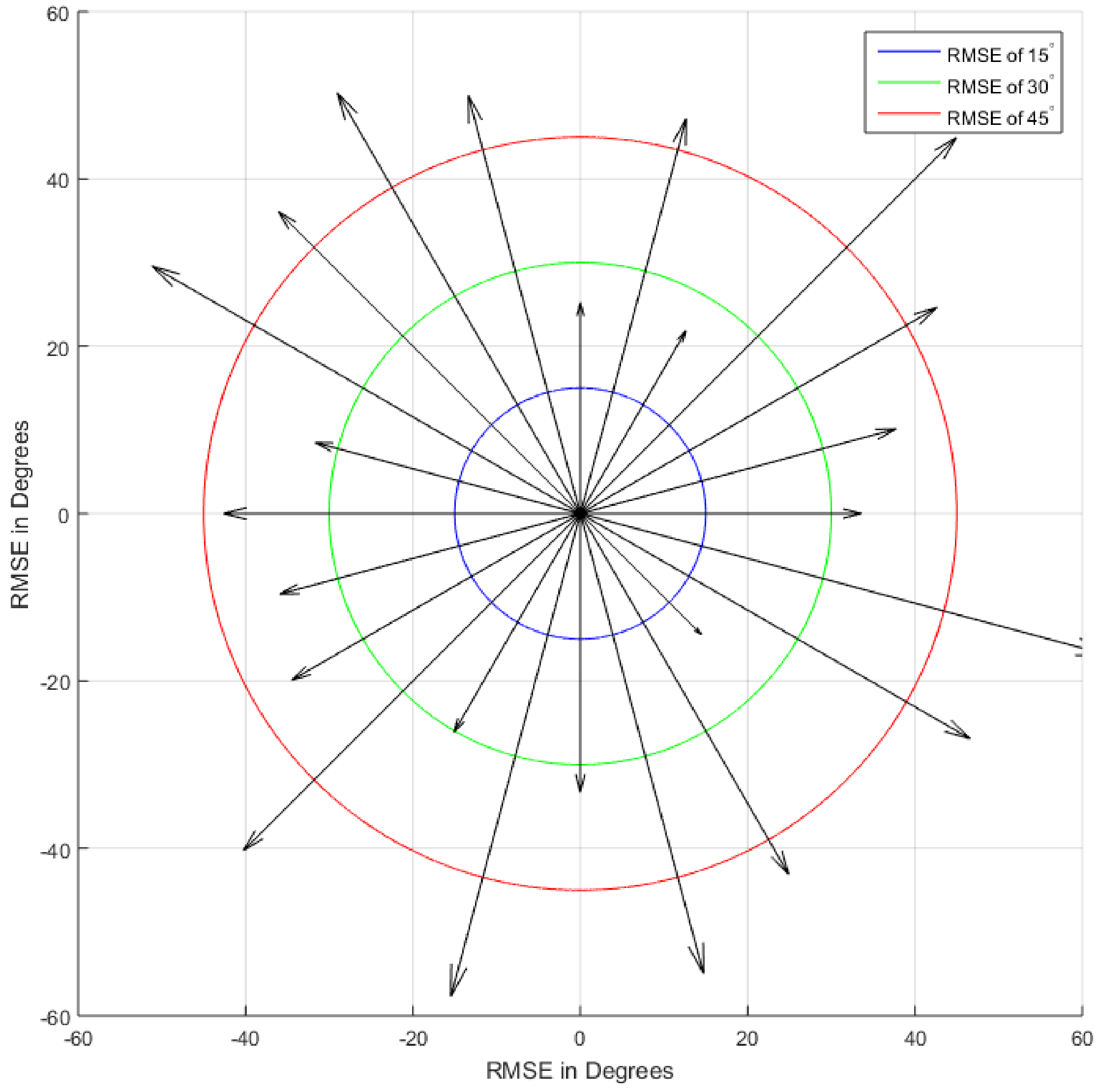

Figure 4.2: Four channel 2D dynamic start angle RSME

Figure 4.3 is a representation of the RMS error (for all 10 participants combined) between the targeted end angle and the perceived end angle as a function of targeted end angle. In Figures 4.2 and 4.3, each vector represents the targeted angle, and the length of the vector corresponds to the RMSE value.

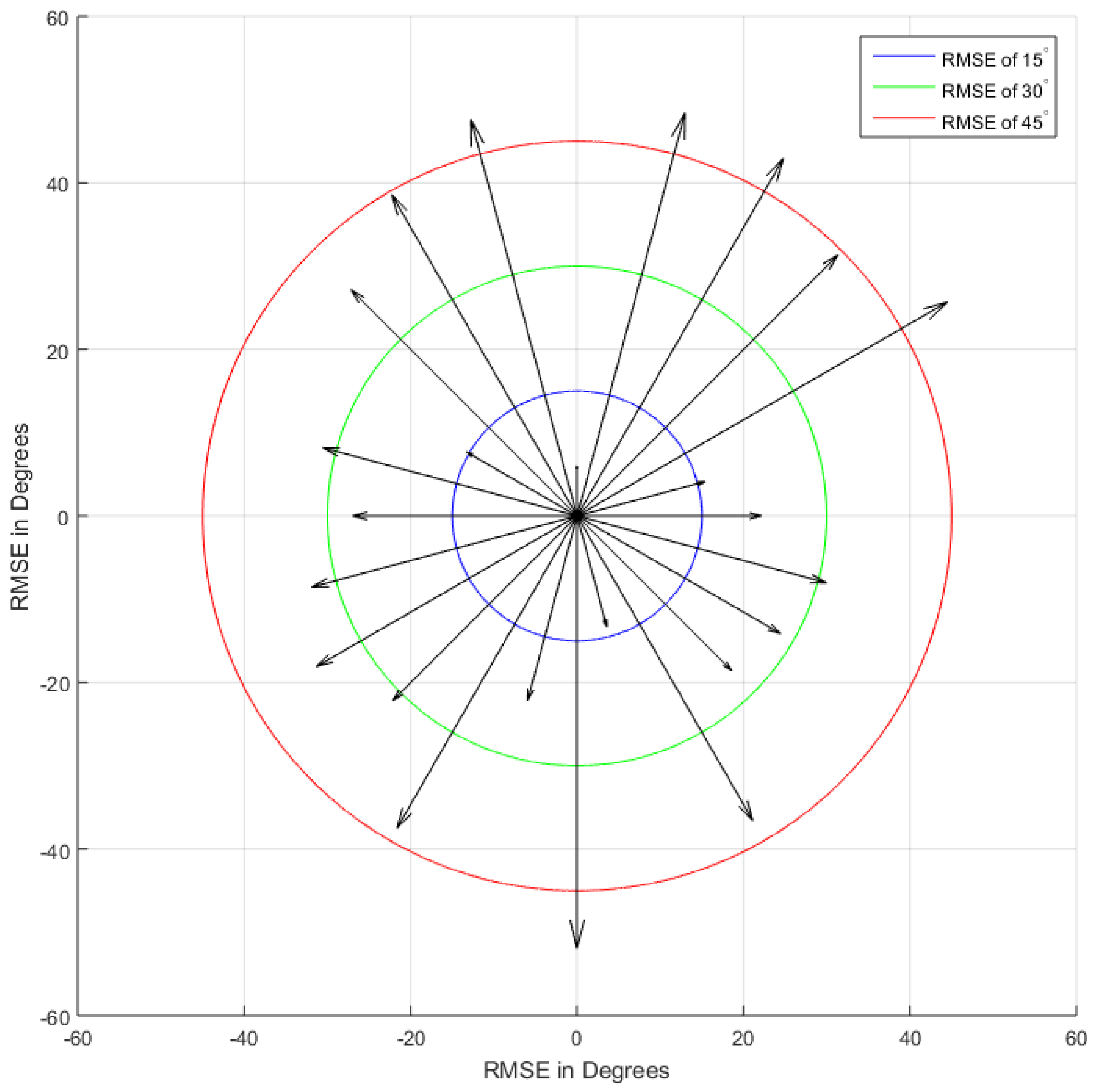

Figure 4.3: Four channel 2D dynamic end angle RSME

The results indicate that for dynamic sounds, the participants had trouble localizing the start of the sound when the tone was calculated to start at an angle diagonally in front or diagonally behind them. The participants had trouble localizing the end of the sound when the tone was calculated to end at an angle diagonally in front or diagonally behind them, and they had trouble localizing the sound when it was calculated to end directly behind them.

4.3 3D Static Sound Localization Test Results

In the four-channel 3D static tests, the participant was given a discrete set of points from which to choose the perceived virtual source position. Figure 4.4 is a bar plot showing the percentage of responses that matched the exact virtual source position. Figure 4.5 is a bar plot showing the percentage of responses that were within 15° of the virtual source position.

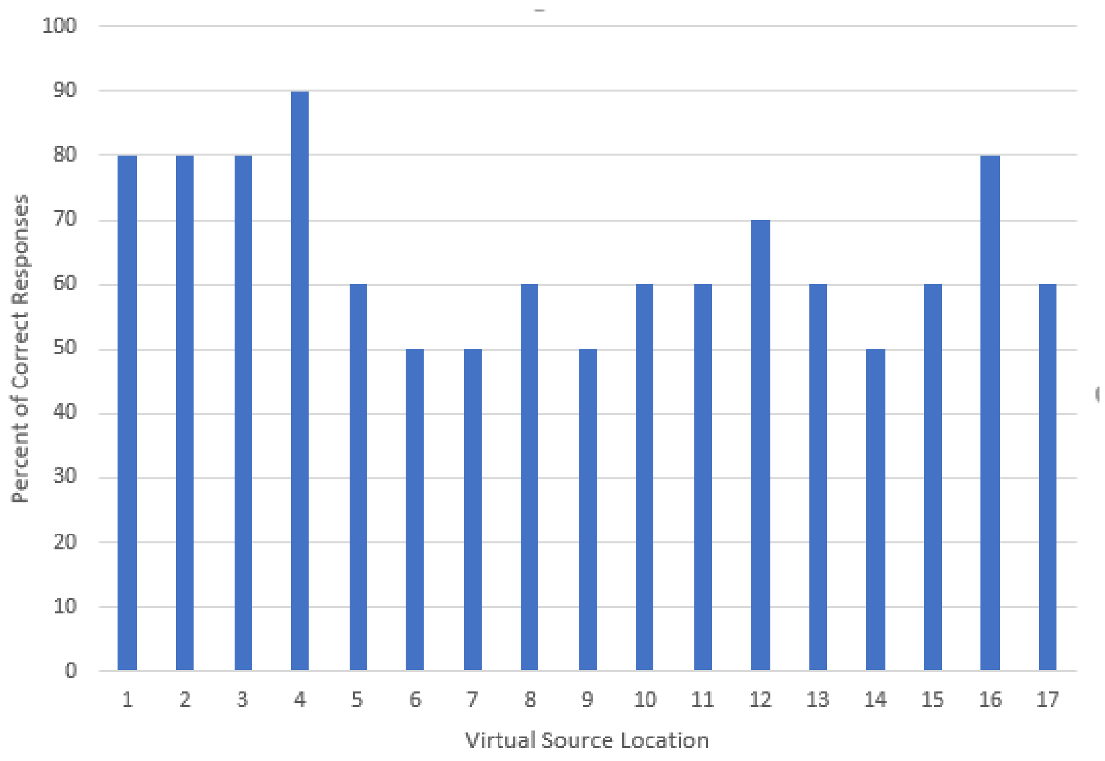

Figure 4.4: Four channel 3D static localization - percent of correct responses

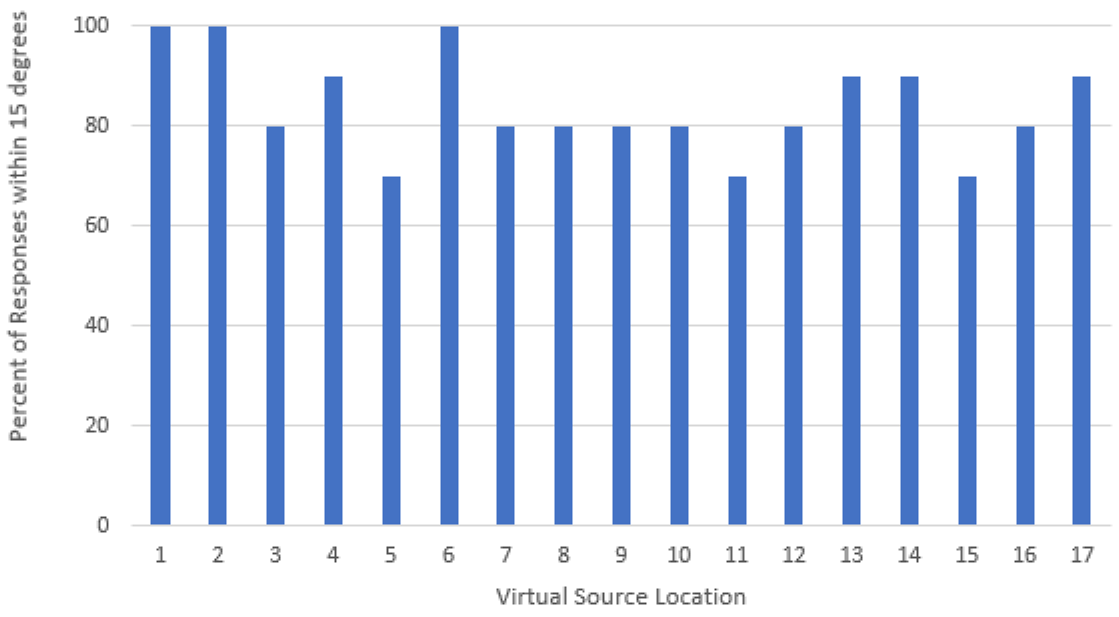

Figure 4.5: Four channel 3D static localization - percent of responses within 15 degree

In these graphs, the sound file number is directly correlated to the virtual source location number shown in Figure 3.7. The results show that individuals perceive the sound well when it is placed on the corners of the diamond shape that is created by the four speakers. When the sound is placed within the diamond, more error occurred.

The RSME was calculated for each of the 17 locations for all participants and the results can be seen in Figure 4.6 where each square represents one of the virtual source locations. For 3D static localization, the average RSME was 15.46°, the minimum RSME was 8.18°, and the maximum RSME was 21.98°.

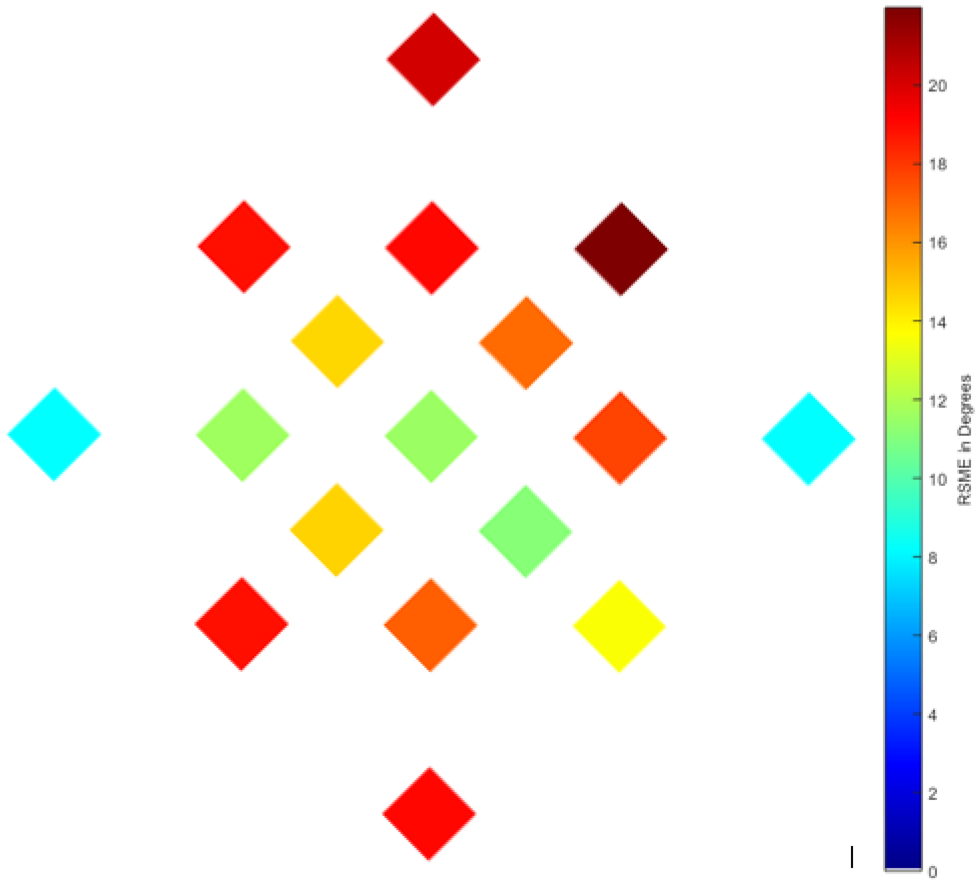

Figure 4.6: RSME for each virtual source location

4.4 3D Dynamic Sound Localization Test Results

In the four channel 3D tests, the participant was given a discrete set of paths from which to choose the perceived virtual source arc. Figure 4.7 is a bar plot showing the percentage of responses that matched the exact theoretical path for each virtual source position by all participants.

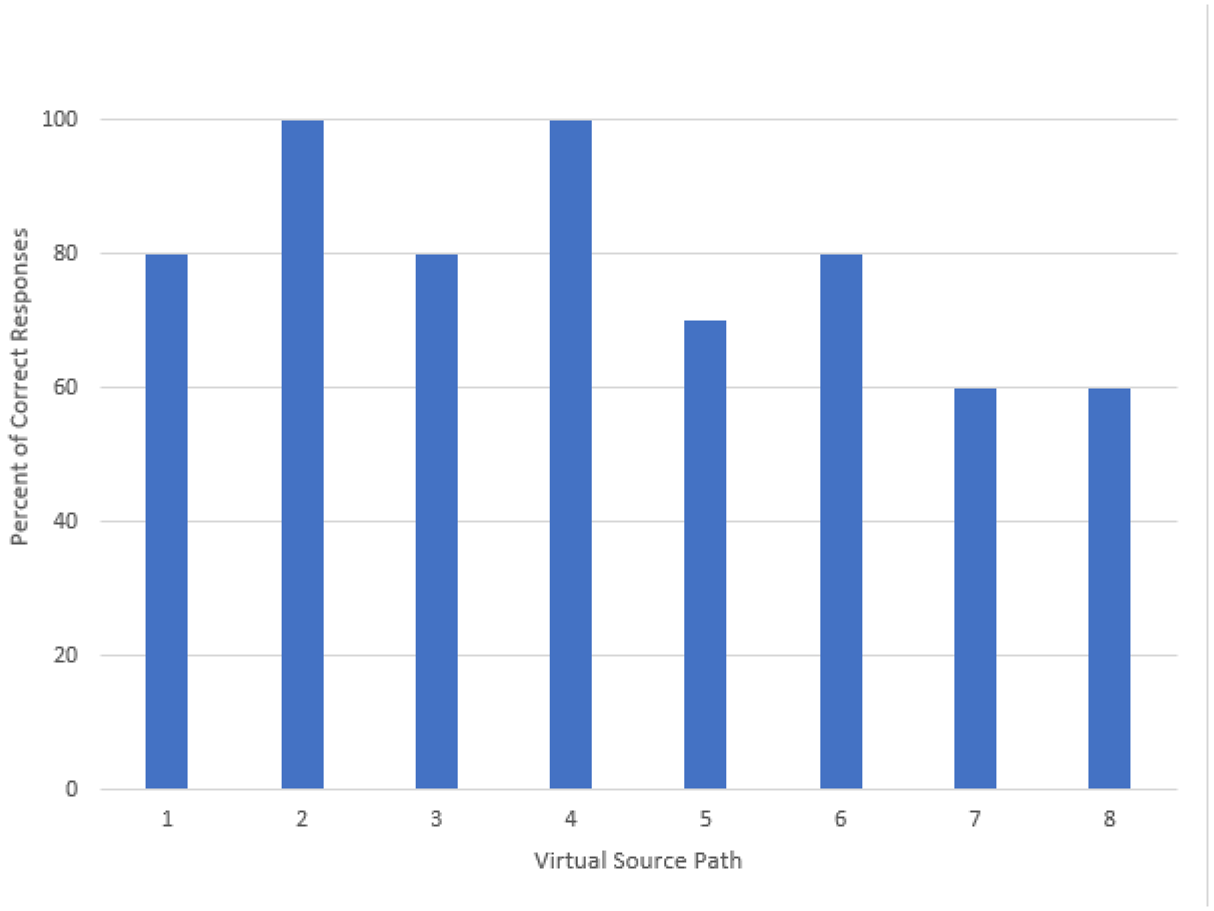

Figure 4.7: Four channel 3D static localization - percent of correct responses

The results show that individuals perceive the sound well when it dynamically travels on the horizontal or vertical plane. When the sound is designed to move diagonally, it causes more error in sound localization. The paths tested are the paths shown in Figure (3.10).

Figure 4.8 is a bar plot showing the percentage of responses that were within one path meaning that the start of the perceived path was with 15° of the theoretical path for each virtual sound arc by all participants.

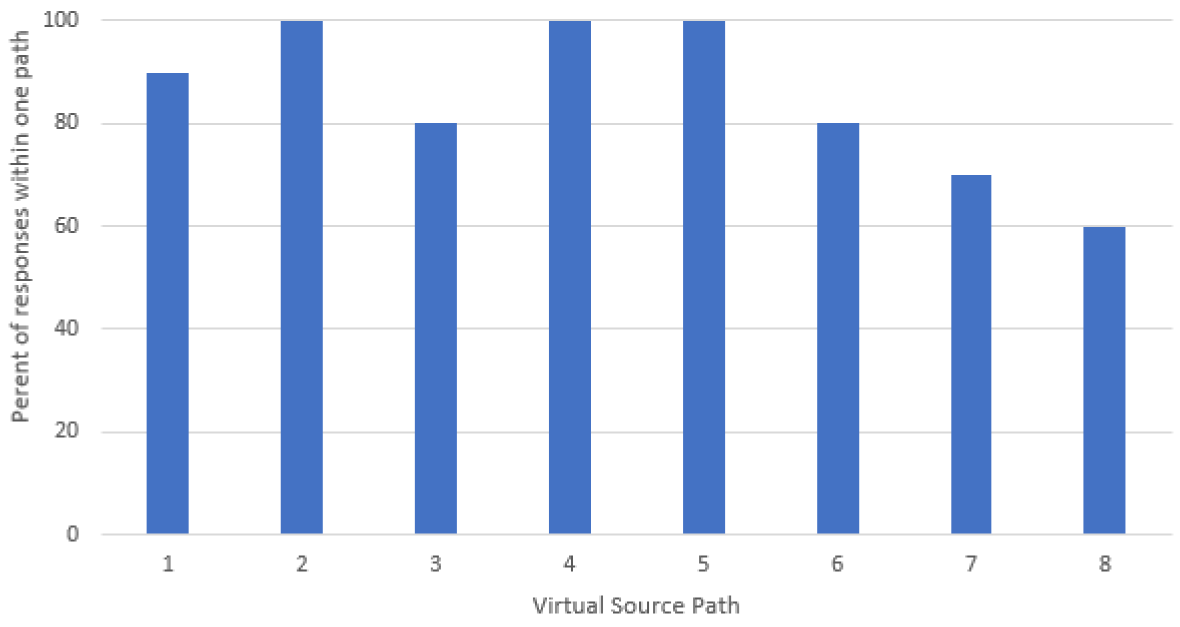

Figure 4.8: Four channel 3D static localization - percent of responses within one path

CHAPTER 5: CONCLUSION AND FUTURE WORK

In this research effort, an algorithm was developed to simulate auditory localization for a four-channel speaker system. Test setups were produced to implement the simulation, and experimental data were collected to verify the simulation. The average RMSE for 2D static sound localization was 37.35°. The average RSME for 2D dynamic sound localization was 50.21° for the calculated start angle and on average the RMSE for the end angle was equal to 47.74°. The static and dynamic testing results seem to indicate that individuals have trouble localizing the source of the sound when it is in the area directly behind them and when the tone is diagonally in front of them or diagonally behind them. Based on these results the simulation is accurate in most areas, but there were some angles where individuals had trouble localizing the sound. Overall the test set up could be useful in implementing a 2D audio system that could provide a more realistic listening experience than current stereo-sound systems.

The average RSME for 3D static localization was 15.4644°. The minimum RSME was 8.18° and the maximum RSME was 21.97°. At high and low elevations, more error occurred compared to an elevation of 0°. This happened because humans are adept at detecting the direction in the horizontal plane due to their ears being placed symmetrically on the head. The ears being symmetric makes it harder to perceive the vertical plane. Overall the test set up could be useful in implementing a 3D audio system that could provide a more realistic listening experience than current stereo-sound systems.

In a future project, the concept of this project could be expanded. For example, the speakers could be moved to different distances and angles from the listener to see if their ability to distinguish location is affected. Other sounds, such as music, speech or sound effects, could also be utilized for testing. Non-equidistant localization could also be explored.

BIBLIOGRAPHY

[1] Daniel M Thompson, Understanding audio: getting the most out of your project or professional recording studio, Hal Leonard Corporation, 2005.

[2] Edward Albert Matthews, Simulation and testing of a multichannel system for 3D sound localization, Western Carolina University, 2015.

[3] Curtis Roads, The computer music tutorial, MIT press, 1996.

[4] Julian Andres Osorio, 20-20 Listening: A sound documentary dedicated to the study of listening experiences in acoustic environments, Ph.D. thesis, Iowa State University, 2016.

[5] Lord Rayleigh, "Xii. on our perception of sound direction," The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 13, no. 74, pp. 214–232, 1907.

[6] William M Hartmann, "How we localize sound," Physics today, vol. 52, pp. 24–29, 1999.

[7] Hans Wallach, Edwin B Newman, and Mark R Rosenzweig, "A precedence effect in sound localization," The Journal of the Acoustical Society of America, vol. 21, no. 4, pp. 468–468, 1949.

[8] M Sawaguchi, "Multichannel sound mixing practice for broadcasting," in IBC Conf. Proc., 1999.

[9] Frederic L Wightman and Doris J Kistler, "Headphone simulation of free-field listening. ii: Psychophysical validation," The Journal of the Acoustical Society of America, vol. 85, no. 2, pp. 868–878, 1989.

[10] Ville Pulkki, "Virtual sound source positioning using vector base amplitude panning," Journal of the audio engineering society, vol. 45, no. 6, pp. 456–466, 1997.

[11] Sascha Spors, Hagen Wierstorf, Alexander Raake, Frank Melchior, Matthias Frank, and

Franz 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, 2013.

[12] Jeroen Breebaart and 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, pp. 1503–1511, 2008.

[13] HAM Clark, GF Dutton, and PB Vanderlyn, “The’stereosonic’recording and reproducing system: A two-channel systems for domestic tape records,” Journal of the Audi Engineering Society, vol. 6, no. 2, pp. 102–117, 1958.

[14] MR Schroeder and BS Atal, “Computer simulation of sound transmission in rooms,” Proceedings of the IEEE, vol. 51, no. 3, pp. 536–537, 1963.

[15] Jens Blauert, Spatial hearing: the psychophysics of human sound localization, MIT press, 1997.

[16] William G Gardner, 3-D audio using loudspeakers, vol. 444, Springer Science & Business Media, 1998.

[17] Myung-Suk Song, Cha Zhang, Dinei Florencio, and Hong-Goo Kang, “An interactive 3-d audio system with loudspeakers,” IEEE Transactions on Multimedia, vol. 13, no. 5, pp. 844–855, 2011.

Appendices

APPENDIX A: COLLECTED DATA

Table A.1 shows the individual responses for each 2D static angle tested across all participants. Table A.2 shows the RSME results of static 2D sound localization testing. Tables A.3 and A.4 shows the individual responses for each dynamic start and end angle tested across all participants. Table A.5 shows the RSME results of dynamic 2D sound localization testing. Table A.6 shows the tested virtual source positions for the 3D static sound localization testing. Table A.7 shows the individual responses for each sound position tested across all participants for 3D static localization. Table A.8 shows the RSME results of static 3D sound localization testing. Table A.9 shows the tested virtual source paths for 3D dynamic sound localization testing. Table A.10 shows the individual responses for each dynamic angle tested across all participants.

Table A.1: 2D static experimental data (all angles in degrees)

	Participant									
Intended Angle	1	2	3	4	5	6	7	8	9	10
0	0	30	15	30	0	0	45	315	0	0
15	0	30	30	30	15	15	330	30	15	15
30	0	345	45	30	270	30	315	30	45	285
45	300	0	45	45	45	0	45	45	270	0
60	285	15	60	60	60	60	60	45	60	255
75	285	60	60	285	120	270	75	90	75	75
90	270	75	90	90	90	90	90	90	105	90
105	225	150	150	255	120	255	105	90	135	105
120	225	240	150	135	135	225	120	120	150	225
135	135	180	135	135	135	225	135	135	135	225
150	180	150	135	135	135	180	150	135	150	135
165	225	195	135	165	150	225	120	135	165	180
180	225	195	150	135	150	225	150	180	180	180
195	225	195	195	180	165	270	150	255	195	180
210	225	180	225	150	135	270	195	240	210	180
225	225	180	225	225	225	270	135	225	225	225
240	225	150	225	255	255	315	150	240	225	225
255	255	105	255	270	270	315	300	255	270	255
270	270	270	90	270	270	270	300	270	270	270
285	300	285	270	285	270	270	270	295	300	315
300	315	30	315	270	300	270	225	315	315	225
315	315	300	315	315	315	300	225	315	315	315
330	345	300	300	285	270	315	0	330	315	0
345	285	270	15	15	330	315	330	300	330	315

Table A.2: Average RSME for each calculated static angle (all angles in degrees)

Intended Angle	RMS Error
0°	24.65°
15°	17.74°
30°	57.12°
45°	49.30°
60°	55.11°
75°	55.72°
90°	6.71°
105°	54.70°
120°	49.52°
135°	42.69°
150°	17.10°
165°	35.18°
180°	30.00°
195°	36.74°
210°	40.25°
225°	34.86°
240°	48.14°
255°	25.54°
270°	57.71°
285°	15.33°
300°	46.96°
315°	29.24°
330°	31.47°
345°	34.53°

Table A.3: 2D dynamic experimental data participants 1 to 5 (all angles in degrees)

		Participant									
Dynamic Start Angle	Dynamic End Angle	1		2		3		4		5	
0	270	300	255	15	285	315	270	45	315	315	225
15	105	315	105	345	105	315	90	30	105	0	90
30	300	60	300	30	105	315	15	45	315	45	315
45	135	45	150	15	105	285	195	45	135	315	135
60	150	90	240	60	180	60	180	15	105	45	135
75	165	90	195	75	165	75	180	45	225	90	240
90	0	90	300	90	0	90	315	90	0	105	45
105	195	105	225	90	180	105	195	90	150	135	225
120	210	120	210	135	180	270	180	135	225	120	210
135	225	195	225	120	195	270	180	135	225	135	225
150	60	210	60	180	60	270	0	195	105	120	30
165	255	225	270	150	75	180	270	150	255	120	210
180	90	225	90	165	90	180	90	180	90	60	150
195	285	195	285	165	255	180	270	165	285	150	240
210	300	225	285	195	285	270	315	225	315	90	270
225	135	225	150	150	240	225	135	225	135	90	135
240	150	240	135	195	135	225	135	225	135	225	135
255	165	255	225	120	195	225	150	270	180	255	135
270	0	270	0	270	0	270	315	270	0	0	270
285	195	270	225	105	195	270	225	240	150	270	195
300	210	315	255	15	240	285	195	330	240	300	210
315	45	315	45	345	240	0	315	315	45	315	45
330	60	330	90	30	120	195	0	300	30	330	60
345	235	300	285	0	270	315	270	45	300	195	270

Table A.4: 2D dynamic experimental data participants 6 to 10 (all angles in degrees)

		Participant									
Dynamic Start Angle	Dynamic End Angle	6		7		8		9		10	
0	270	0	270	0	270	315	45	315	225	0	270
15	105	315	225	330	60	45	105	315	105	0	120
30	300	30	300	300	210	45	315	45	315	270	0
45	135	315	225	30	120	45	135	45	135	270	180
60	150	60	315	60	150	45	135	30	120	0	90
75	165	75	165	75	195	45	135	75	180	270	180
90	0	135	225	90	0	45	315	150	0	90	0
105	195	270	225	105	210	135	225	105	225	45	135
120	210	120	315	135	180	135	225	120	210	255	165
135	225	225	315	135	225	135	225	135	225	180	90
150	60	150	60	285	195	135	45	120	30	150	60
165	255	165	225	240	225	135	225	150	240	180	270
180	90	180	90	195	105	120	30	135	225	180	90
195	285	180	270	285	285	135	45	225	315	180	270
210	300	225	315	210	300	225	315	225	315	225	315
225	135	315	225	225	135	225	135	225	135	135	225
240	150	240	150	240	150	225	135	225	135	150	240
255	165	225	315	240	150	45	135	225	135	240	150
270	0	270	0	330	0	315	45	270	300	270	315
285	195	285	195	285	195	315	225	330	240	240	150
300	210	180	270	330	240	315	225	300	210	30	120
315	45	315	45	300	210	315	45	315	315	0	90
330	60	225	60	300	60	315	45	330	60	300	60
345	235	135	270	330	240	330	45	315	225	0	90

Table A.5: Average RSME for each calculated dynamic start and dynamic end angle (all angles in degrees)

Intended Start Angle	RMS Error	Intended End Angle	RMS Error
0°	37.35°	270°	49.75°
15°	43.37°	105°	41.35°
30°	54.70°	300°	67.42°
45°	70.68°	135°	38.83°
60°	28.06°	150°	66.41°
75°	54.29°	165°	35.50°
90°	28.06°	0°	52.82°
105°	57.51°	195°	32.52°
120°	64.52°	210°	39.62°
135°	56.72°	225°	54.08°
150°	65.73°	60°	50.87°
165°	36.43°	255°	60.19°
180°	47.43°	90°	50.64°
195°	41.35°	285°	43.47°
210°	44.24°	300°	16.43°
225°	63.29°	135°	52.39°
240°	33.54°	150°	31.10°
255°	66.41°	165°	55.32°
270°	37.05°	0°	42.16°
285°	63.29°	195°	29.62°
300°	55.32°	210°	41.08°
315°	22.76°	45°	85.25°
330°	59.81°	60°	30.37°
345°	73.02°	235°	78.53°

Table A.6: 3D random angle test position (all angles in degrees)

Random Angle Position	Azimuth	Elevation
1	30	0
2	-30	0
3	0	30
4	0	-30
5	0	0
6	15	0
7	-15	0
8	0	15
9	0	-15
10	7.5	7.5
11	-7.5	7.5
12	7.5	-7.5
13	-7.5	-7.5
14	15	15
15	-15	15
16	15	-15
17	-15	15

Table A.7: 3D static localization experimental data

	Participant									
Sound Location	1	2	3	4	5	6	7	8	9	10
1	16	1	1	6	1	1	1	1	1	1
2	15	2	7	2	2	2	2	2	2	2
3	3	3	9	3	3	3	3	3	9	3
4	3	4	4	4	4	4	4	4	4	4
5	5	5	8	17	5	5	5	17	7	5
6	1	6	14	6	1	6	1	6	1	6
7	7	7	3	2	7	7	2	7	2	6
8	8	8	4	9	8	8	5	3	8	8
9	9	9	6	5	4	9	9	5	8	9
10	10	10	1	10	6	10	14	1	10	10
11	11	11	3	11	11	11	17	2	11	10
12	10	12	1	16	12	12	12	12	12	12
13	13	13	13	17	13	13	2	17	7	13
14	1	14	14	1	14	14	16	14	6	13
15	17	15	2	15	15	1	15	15	15	14
16	14	16	16	16	16	16	16	16	16	2
17	11	17	17	2	17	17	13	17	13	17

Table A.8: Average RSME for each 3D static location

Sound Location	RMS Error
1	8.18
2	8.18
3	20.12
4	18.97
5	11.57
6	11.61
7	17.68
8	18.97
9	17.08
10	14.56
11	16.84
12	14.67
13	11.11
14	18.83
15	21.97
16	18.83
17	13.64

Table A.9: 3D random angle test positions for the start and end angles

Start Angle Location	Azimuth	Elevation	End Angle Location	Azimuth	Elevation
1	30	0	2	-30	0
2	-30	0	1	30	0
3	0	30	4	0	-30
4	0	-30	3	0	30
5	15	15	7	-15	-15
6	15	-15	8	-15	15
7	-15	-15	5	15	15
8	-15	15	6	15	-15

Table A.10: 3D dynamic localization experimental data

	Participant									
Dynamic Path	1	2	3	4	5	6	7	8	9	10
1	6	1	1	2	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2
3	3	1	3	3	3	3	3	4	3	3
4	4	4	4	4	4	4	4	4	4	4
5	1	5	5	1	5	5	5	1	5	5
6	5	6	6	6	6	6	6	6	6	5
7	8	7	7	7	7	7	8	2	7	5
8	8	7	8	8	8	8	7	7	8	6

APPENDIX B: MATLAB CODE

2D Static and Dynamic Localization

```
% Clears all previous data and clears workspace
clear all;
close all;
clc;

% Speaker locations in terms of azimuth (angle)
Speaker1Theta=45;  % Speaker 1 location
Speaker2Theta=135; % Speaker 2 location
Speaker3Theta=225; % Speaker 3 location
Speaker4Theta=315; % Speaker 4 location
Listener= [0 0];    % The observers position
s=6;  % How far away all the speakers are from the Listener (in feet)

%Angles used for random testing
Theta= [15 45 330 225 30 105 195 345 60 180 240 300 150 0 270 315 135 165 90 75 285 120
210 255]

% This section is for static tones
% Generates tones used in testing
[Tone, FS, t] =Tone_Generator (400,1,2,0,0);

i=1;
n=1;

% The following if loop calculates the gains for the speakers
if Theta(i)==Speaker1Theta
        g1=1;
        g2=0;
        g3=0;
        g4=0;
        XLeft=s*cosd(Speaker1Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker1Theta); %Left speaker coordinate
        px=s*cosd(Theta(i));
        py=s*sind(Theta(i));
        figure
        plot ([0 XLeft], [0 YLeft],'r-o','linewidth',2)
        hold on
        plot ([0 px], [0 py],'k-x')
        legend ('Speaker’, ‘Virtual Source')
        xlabel ('Distance in Feet')
```

```
            ylabel ('Distance in Feet')
            title ('Speaker Locations and Virtual Source Location')
            grid on
            axis ([-7 7 -7 7])
            axis('square')
    elseif Theta(i)==Speaker2Theta
            g1=0;
            g2=1;
            g3=0;
            g4=0;
            XLeft=s*cosd(Speaker2Theta);  %Left speaker coordinate
            YLeft=s*sind(Speaker2Theta); %Left speaker coordinate
            px=s*cosd(Theta(i));
            py=s*sind(Theta(i));
            figure
            plot ([0 XLeft], [0 YLeft],'r-o','linewidth',2)
            hold on
            plot ([0 px], [0 py],'k-x')
            legend ('Speaker’, ‘Virtual Source')
            xlabel ('Distance in Feet')
            ylabel ('Distance in Feet')
            title ('Speaker Locations and Virtual Source Location')
            grid on
            axis ([-7 7 -7 7])
            axis('square')
    elseif Theta(i)==Speaker3Theta
            g1=0;
            g2=0;
            g3=1;
            g4=0;
            XLeft=s*cosd(Speaker3Theta);  %Left speaker coordinate
            YLeft=s*sind(Speaker3Theta); %Left speaker coordinate
            px=s*cosd(Theta(i));
            py=s*sind(Theta(i));
            figure
            plot ([0 XLeft], [0 YLeft],'r-o','linewidth',2)

            hold on


            legend ('Speaker’, ‘Virtual Source')
            xlabel ('Distance in Feet')
            ylabel ('Distance in Feet')
            title ('Speaker Locations and Virtual Source Location')
            grid on
            axis ([-7 7 -7 7])
```

```
        axis('square')
    elseif Theta(i)==Speaker4Theta
        g1=0;
        g2=0;
        g3=0;
        g4=1;
        XLeft=s*cosd(Speaker4Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker4Theta); %Left speaker coordinate
        px=s*cosd(Theta(i));
        py=s*sind(Theta(i));
        figure
        plot ([0 XLeft], [0 YLeft],'r-o','linewidth',2)
        hold on
        plot ([0 px], [0 py],'k-x')
        legend ('Speaker', 'Virtual Source')
        xlabel ('Distance in Feet')
        ylabel ('Distance in Feet')
        title ('Speaker Locations and Virtual Source Location')
        grid on
        axis ([-7 7 -7 7])
        axis('square')
    elseif Theta(i)>Speaker1Theta && Theta(i)<Speaker2Theta
        XLeft=s*cosd(Speaker2Theta-90);  %Left speaker coordinate
        YLeft=s*sind(Speaker2Theta-90); %Left speaker coordinate
        XRight=s*cosd(Speaker1Theta-90); %Right speaker coordinate
        YRight=s*sind(Speaker1Theta-90); %Right speaker coordinate
        Theta(i)=Theta(i)-90;
        [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta(i));
        g1=gCorrect (2);
        g2=gCorrect (1);
        g3=0;
        g4=0;
        XLeft=s*cosd(Speaker2Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker2Theta); %Left speaker coordinate
        XRight=s*cosd(Speaker1Theta); %Right speaker coordinate
        YRight=s*sind(Speaker1Theta); %Right speaker coordinate
        px=s*cosd(Theta(i)+90);
        py=s*sind(Theta(i)+90);
        figure
        plot ([0 XLeft], [0 YLeft],'r-o')
        hold on
        plot ([0 XRight], [0 YRight],'-o')
        plot ([0 px], [0 py],'k-x')
        legend ('Left Speaker', 'Right Speaker', 'Virtual Source')
        xlabel ('Distance in Feet')
        ylabel ('Distance in Feet')
        title ('Speaker Locations and Virtual Source Location')
        grid on
        axis ([-7 7 -7 7])
        axis('square')
        strl=sprintf ('Left Speaker Gain=%f’, gCorrect (1));
        strr=sprintf ('Right Speaker Gain=%f’, gCorrect (2));
        text (-6, -6, strl)
        text (-6, -5.5, strr)
elseif Theta(i)>Speaker2Theta && Theta(i)<Speaker3Theta
        XLeft=s*cosd(Speaker3Theta-180);  %Left speaker coordinate
        YLeft=s*sind(Speaker3Theta-180); %Left speaker coordinate
        XRight=s*cosd(Speaker2Theta-180); %Right speaker coordinate
        YRight=s*sind(Speaker2Theta-180); %Right speaker coordinate
        Theta(i)=Theta(i)-180;
        [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta(i));
        g1=0;
        g2=gCorrect (2);
        g3=gCorrect (1);
        g4=0;
        XLeft=s*cosd(Speaker3Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker3Theta);  %Left speaker coordinate
        XRight=s*cosd(Speaker2Theta); %Right speaker coordinate
        YRight=s*sind(Speaker2Theta); %Right speaker coordinate
        px=s*cosd(Theta(i)+180);
        py=s*sind(Theta(i)+180);
        figure
        plot ([0 XLeft], [0 YLeft],'r-o')
        hold on
        plot ([0 XRight], [0 YRight],'-o')
        plot ([0 px], [0 py],'k-x')
        legend ('Left Speaker’, ‘Right Speaker’, ‘Virtual Source')
        xlabel ('Distance in Feet')
        ylabel ('Distance in Feet')
        title ('Speaker Locations and Virtual Source Location')
        grid on
        axis ([-7 7 -7 7])
        axis('square')
        strl=sprintf ('Left Speaker Gain=%f’, gCorrect (1));
        strr=sprintf ('Right Speaker Gain=%f’, gCorrect (2));
        text (-6, -6, strl)
        text (-6, -5.5, strr)
elseif Theta(i)>Speaker3Theta && Theta(i)<Speaker4Theta
        XLeft=s*cosd(Speaker4Theta-270);  %Left speaker coordinate
        YLeft=s*sind(Speaker4Theta-270);  %Left speaker coordinate
        XRight=s*cosd(Speaker3Theta-270); %Right speaker coordinate
        YRight=s*sind(Speaker3Theta-270); %Right speaker coordinate
        Theta(i)=Theta(i)-270;
        [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta(i));
        g1=0;
        g2=0;
        g3=gCorrect (2);
        g4=gCorrect (1);
        XLeft=s*cosd(Speaker4Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker4Theta);  %Left speaker coordinate
        XRight=s*cosd(Speaker3Theta); %Right speaker coordinate
        YRight=s*sind(Speaker3Theta); %Right speaker coordinate
        px=s*cosd(Theta(i)+270);
        py=s*sind(Theta(i)+270);
        figure
        plot ([0 XLeft], [0 YLeft],'r-o')
        hold on
        plot ([0 XRight], [0 YRight],'-o')
        plot ([0 px], [0 py],'k-x')
        legend ('Left Speaker', 'Right Speaker', 'Virtual Source')
        xlabel ('Distance in Feet')
        ylabel ('Distance in Feet')
        title ('Speaker Locations and Virtual Source Location')
        grid on
        axis ([-7 7 -7 7])
        axis('square')
        strl=sprintf ('Left Speaker Gain=%f', gCorrect (1));
        strr=sprintf ('Right Speaker Gain=%f', gCorrect (2));
        text (-6, -6, strl)
        text (-6, -5.5, strr)
    else
        XLeft=s*cosd(Speaker1Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker1Theta);  %Left speaker coordinate
        XRight=s*cosd(Speaker4Theta); %Right speaker coordinate
        YRight=s*sind(Speaker4Theta); %Right speaker coordinate
        [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta(i));
        g1=gCorrect (1);
        g2=0;
        g3=0;
        g4=gCorrect (2);
        XLeft=s*cosd(Speaker1Theta);  %Left speaker coordinate
        YLeft=s*sind(Speaker1Theta); %Left speaker coordinate
        XRight=s*cosd(Speaker4Theta); %Right speaker coordinate
        YRight=s*sind(Speaker4Theta); %Right speaker coordinate
        px=s*cosd(Theta(i));
        py=s*sind(Theta(i));
        figure
        plot ([0 XLeft], [0 YLeft],'r-o')
        hold on
        plot ([0 XRight], [0 YRight],'-o')
        plot ([0 px], [0 py],'k-x')
        legend ('Left Speaker', 'Right Speaker', 'Virtual Source')
        xlabel ('Distance in Feet')
        ylabel ('Distance in Feet')
        title ('Speaker Locations and Virtual Source Location')
        grid on
        axis ([-7 7 -7 7])
        axis('square')
        strl=sprintf ('Left Speaker Gain=%f', gCorrect (1));
        strr=sprintf ('Right Speaker Gain=%f', gCorrect (2));
        text (-6, -6, strl)
        text (-6, -5.5, strr)
end
Y(:,2) =Tone*g1; %Channel 1
Y(:,1) =Tone*g2; %Channel 2
Y(:,4) =Tone*g3; %Channel 3
Y(:,3) =Tone*g4; %Channel 4
audiowrite ('StaticTestingTheta24.wav', Y, FS);

Theta= [120 45 300 15 225 180 135 330 60 345 195 150 255 165 30 210 270 90 0 315 240 105
285 75]
theta_1=Theta(n);    % Theta 1
theta_2=theta_1+90;  % Theta 2

% Used to control the bounds of theta
if theta_2>360
   theta_1=theta_1-360;
   theta_2=theta_2-360;
   ActualEnd=theta_2;
end
if theta_2<0
   theta_1=theta_1+360;
   theta_2=theta_2+360;
   ActualEnd=theta_2;
end

%Used to allow the code to go from to 360 back to 0
if theta_1>=225 && theta_2<=45
   theta_1=theta_1-360;
end
if theta_2>=225 && theta_1<=45
   theta_2=theta_2-360;
```

```
end

[Tone, FS, t] =Tone_Generator(400,1,2,0,0);
ThetaDynamic=linspace (theta_1, theta_2,100); %Range of Theta from theta 1 to theta 2
TS=1/FS; %Period of the .wav file
figure
X1=s*cosd(Speaker1Theta);
Y1=s*sind(Speaker1Theta);
X2=s*cosd(Speaker2Theta);
Y2=s*sind(Speaker2Theta);
X3=s*cosd(Speaker3Theta);
Y3=s*sind(Speaker3Theta);
X4=s*cosd(Speaker4Theta);
Y4=s*sind(Speaker4Theta);
plot ([0, X1], [0, Y1],'r-o')
hold on
plot ([0, X2], [0, Y2],'b-o')
plot ([0, X3], [0, Y3],'g-o')
plot ([0, X4], [0, Y4],'k-o')
xlabel ('Distance in Feet')
ylabel ('Distance in Feet')
title ('Speaker Locations and Virtual Source Location')
axis ([-7 7 -7 7])
axis('square')
grid on
for i=1: length(ThetaDynamic)
        if ThetaDynamic(i)<0
                ThetaDynamic(i)=ThetaDynamic(i)+360;
        end
        if ThetaDynamic(i)>360
                ThetaDynamic(i)=ThetaDynamic(i)-360;
        end
   % The following if loop calculates the gains for the speakers
        if ThetaDynamic(i)==Speaker1Theta
                g1(i)=1;
                g2(i)=0;
                g3(i)=0;
                g4(i)=0;
                px=s*cosd(ThetaDynamic(i));
                py=s*sind(ThetaDynamic(i));
                plot (px, py, ‘kx')
                hold on
        elseif ThetaDynamic(i)==Speaker2Theta
                g1(i)=0;
                g2(i)=1;
                g3(i)=0;
```

```
          g4(i)=0;
          px=s*cosd(ThetaDynamic(i));
          py=s*sind(ThetaDynamic(i));
          plot (px, py, ‘kx')
          hold on
elseif ThetaDynamic(i)==Speaker3Theta
          g1(i)=0;
          g2(i)=0;
          g3(i)=1;
          g4(i)=0;
          px=s*cosd(ThetaDynamic(i));
          py=s*sind(ThetaDynamic(i));
          plot (px, py, ‘kx')
          hold on
elseif ThetaDynamic(i)==Speaker4Theta
          g1(i)=0;
          g2(i)=0;
          g3(i)=0;
          g4(i)=1;
          px=s*cosd(ThetaDynamic(i));
          py=s*sind(ThetaDynamic(i));
          plot (px, py, ‘kx')
          hold on
elseif ThetaDynamic(i)>Speaker1Theta && ThetaDynamic(i)<Speaker2Theta
          XLeft=s*cosd(Speaker2Theta-90);  %Left speaker coordinate
          YLeft=s*sind(Speaker2Theta-90);  %Left speaker coordinate
           XRight=s*cosd(Speaker1Theta-90); %Right speaker coordinate
           YRight=s*sind(Speaker1Theta-90); %Right speaker coordinate
           ThetaNew=ThetaDynamic(i)-90;
           [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, ThetaNew);
           g1(i)=gCorrect (2);
           g2(i)=gCorrect (1);
           g3(i)=0;
           g4(i)=0;
           px=s*cosd(ThetaDynamic(i));
           py=s*sind(ThetaDynamic(i));
           plot (px, py, ‘kx')
           hold on
elseif ThetaDynamic(i)>Speaker2Theta && ThetaDynamic(i)<Speaker3Theta
           XLeft=s*cosd(Speaker3Theta-180);  %Left speaker coordinate
           YLeft=s*sind(Speaker3Theta-180);  %Left speaker coordinate
           XRight=s*cosd(Speaker2Theta-180); %Right speaker coordinate
           YRight=s*sind(Speaker2Theta-180); %Right speaker coordinate
           ThetaNew=ThetaDynamic(i)-180;
           [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, ThetaNew);
           g1(i)=0;
```

```
                    g2(i)=gCorrect (2);
                    g3(i)=gCorrect (1);
                    g4(i)=0;
                    px=s*cosd(ThetaDynamic(i));
                    py=s*sind(ThetaDynamic(i));
                    plot (px, py, ‘kx')
                    hold on
          elseif ThetaDynamic(i)>Speaker3Theta && ThetaDynamic(i)<Speaker4Theta
                    XLeft=s*cosd(Speaker4Theta-270);  %Left speaker coordinate
                    YLeft=s*sind(Speaker4Theta-270);  %Left speaker coordinate
                    XRight=s*cosd(Speaker3Theta-270); %Right speaker coordinate
                    YRight=s*sind(Speaker3Theta-270); %Right speaker coordinate
                    ThetaNew=ThetaDynamic(i)-270;
                    [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, ThetaNew);
                    g1(i)=0;
                    g2(i)=0;
                    g3(i)=gCorrect (2);
                    g4(i)=gCorrect (1);
                    px=s*cosd(ThetaDynamic(i));
                    py=s*sind(ThetaDynamic(i));
                    plot (px, py, ‘kx')
                    hold on
          else
                    XLeft=s*cosd(Speaker1Theta);  %Left speaker coordinate
                    YLeft=s*sind(Speaker1Theta);  %Left speaker coordinate
                    XRight=s*cosd(Speaker4Theta); %Right speaker coordinate
                    YRight=s*sind(Speaker4Theta); %Right speaker coordinate
                    [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight,
                    ThetaDynamic(i));
                    g1(i)=gCorrect (1);
                    g2(i)=0;
                    g3(i)=0;
                    g4(i)=gCorrect (2);
                    px=s*cosd(ThetaDynamic(i));
                    py=s*sind(ThetaDynamic(i));
                    plot (px, py, ‘kx')
                    hold on
          end
end

legend ('First Speaker’, ‘Second Speaker’, ‘Third Speaker’, ‘Fourth Speaker’, ‘Virtual Source')
axis ([-7 7 -7 7])
axis('square')
vq1=interp1(linspace (0, max(t), length(g1)), g1, t);
vq2=interp1(linspace (0, max(t), length(g2)), g2, t);
vq3=interp1(linspace (0, max(t), length(g3)), g3, t);
```

```
vq4=interp1(linspace (0, max(t), length(g4)), g4, t);
Y(:,2) =Tone. *vq1; %Channel 1
Y(:,1) =Tone. *vq2; %Channel 2
Y(:,4) =Tone. *vq3; %Channel 3
Y(:,3) =Tone. *vq4; %Channel 4
audiowrite ('DynamicTestingTheta24.wav’, Y, FS);
```

3D Static Localization

```
close all;
clear all;
clc;

s=6;

%Speaker Locations in terms of degrees
Speaker1_azumith=30;
Speaker1_elavation=0;
Speaker2_azumith=-30;
Speaker2_elavation=0;
Speaker3_azumith=0;
Speaker3_elavation=30;
Speaker4_azumith=0;
Speaker4_elavation=-30;

Phantom_desired_azumith=-15
Phantom_desired_elavation=-10

if Phantom_desired_elavation>0 & Phantom_desired_azumith~=0 %Upper triangle
        L1z=s*sind(Speaker1_elavation);
        d=sqrt(s^2-L1z^2)
        L1x=d*cosd(Speaker1_azumith);
        L1y=d*sind(Speaker1_azumith);
        L2z=s*sind(Speaker2_elavation);
        d=sqrt(s^2-L2z^2)
        L2x=d*cosd(Speaker2_azumith);
        L2y=d*sind(Speaker2_azumith);
        L3z=s*sind(Speaker3_elavation);
        d=sqrt(s^2-L3z^2)
        L3x=d*cosd(Speaker3_azumith);
        L3y=d*sind(Speaker3_azumith);
        L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
        px=s*cosd(Phantom_desired_azumith);
        py=s*sind(Phantom_desired_azumith);
        pz=sqrt(px^2+py^2) *tand(Phantom_desired_elavation);
        p= [px py pz];
        g=p*inv(L)
        if max(g)>1
                n=sqrt (g (1) ^2+g (2) ^2+g (3) ^2)
                g=g/n
        end
        g1=g (1);     %speaker 1 gain
        g2=g (2);      %speaker 2 gain
        g3=g (3);    %speaker 3 gain
        g4=0;        %speaker 4 gain
elseif Phantom_desired_elavation<0 & Phantom_desired_azumith~=0 %lower triangle
        L1z=s*sind(Speaker1_elavation);
        d=sqrt(s^2-L1z^2)
        L1x=d*cosd(Speaker1_azumith);
        L1y=d*sind(Speaker1_azumith);
        L2z=s*sind(Speaker2_elavation);
        d=sqrt(s^2-L2z^2)
        L2x=d*cosd(Speaker2_azumith);
        L2y=d*sind(Speaker2_azumith);
        L3z=s*sind(abs(Speaker4_elavation));
        d=sqrt(s^2-L3z^2)
        L3x=d*cosd(abs(Speaker4_azumith));
        L3y=d*sind(abs(Speaker4_azumith));
        L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
        px=s*cosd(Phantom_desired_azumith);
        py=s*sind(Phantom_desired_azumith);
        pz=sqrt(px^2+py^2) *tand(abs(Phantom_desired_elavation));
        p= [px py pz];
        g=p*inv(L)
        if max(g)>1
                n=sqrt (g (1) ^2+g (2) ^2+g (3) ^2)
                g=g/n
        end
        g1=g (1);     %speaker 1 gain
        g2=g (2);     %speaker 2 gain
        g3=0;         %speaker 3 gain
        g4=g (3);     %speaker 4 gain
        % Make certain z negative for graphing purposes
        L3z=-s*sind(abs(Speaker4_elavation));
        pz=-sqrt(px^2+py^2) *tand(abs(Phantom_desired_elavation));
elseif Phantom_desired_elavation==0 & Phantom_desired_azumith~=0 %Midspeakers
        XLeft=s*cosd(Speaker1_azumith);  %Left speaker coordinate
        YLeft=s*sind(Speaker1_azumith);  %Left speaker coordinate
        XRight=s*cosd(Speaker2_azumith); %Right speaker coordinate
        YRight=s*sind(Speaker2_azumith); %Right speaker coordinate
        Theta=Phantom_desired_azumith
        [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta);
        g1=gCorrect (1);     %speaker 1 gain
        g2=gCorrect (2);     %speaker 2 gain
        g3=0;                %speaker 3 gain
        g4=0;                %speaker 4 gain
        L1z=s*sind(Speaker1_elavation);
        d=sqrt(s^2-L1z^2)
        L1x=d*cosd(Speaker1_azumith);
```

```
          L1y=d*sind(Speaker1_azumith);
          L2z=s*sind(Speaker2_elavation);
          d=sqrt(s^2-L2z^2)
          L2x=d*cosd(Speaker2_azumith);
          L2y=d*sind(Speaker2_azumith);
          L3z=s*sind(Speaker3_elavation);
          d=sqrt(s^2-L3z^2)
          L3x=d*cosd(Speaker3_azumith);
          L3y=d*sind(Speaker3_azumith);
          L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
          px=s*cosd(Phantom_desired_azumith);
          py=s*sind(Phantom_desired_azumith);
          pz=sqrt(px^2+py^2) *tand(Phantom_desired_elavation);
          p= [px py pz];
elseif Phantom_desired_elavation~=0 & Phantom_desired_azumith==0 %top and bottom
speaker
          XLeft=s*cosd(Speaker3_elavation);  %Left speaker coordinate
          YLeft=s*sind(Speaker3_elavation);  %Left speaker coordinate
          XRight=s*cosd(Speaker4_elavation); %Right speaker coordinate
          YRight=s*sind(Speaker4_elavation); %Right speaker coordinate
          Theta=Phantom_desired_elavation
          [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta);
          g1=0;              %speaker 1 gain
          g2=0;              %speaker 2 gain
          g3=gCorrect (1);      %speaker 3 gain
          g4=gCorrect (2);      %speaker 4 gain
          L1z=s*sind(Speaker1_elavation);
          d=sqrt(s^2-L1z^2)
          L1x=d*cosd(Speaker1_azumith);
          L1y=d*sind(Speaker1_azumith);
          L2z=s*sind(Speaker2_elavation);
          d=sqrt(s^2-L2z^2)
          L2x=d*cosd(Speaker2_azumith);
          L2y=d*sind(Speaker2_azumith);
          L3z=s*sind(Speaker3_elavation);
          d=sqrt(s^2-L3z^2)
          L3x=d*cosd(Speaker3_azumith);
          L3y=d*sind(Speaker3_azumith);
          L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
          px=s*cosd(Phantom_desired_azumith);
          py=s*sind(Phantom_desired_azumith);
          pz=sqrt(px^2+py^2) *tand(Phantom_desired_elavation);
          p= [px py pz];
elseif Phantom_desired_elavation==0 & Phantom_desired_azumith==0 % Midspeakers
          XLeft=s*cosd(Speaker1_azumith);  %Left speaker coordinate
          YLeft=s*sind(Speaker1_azumith);  %Left speaker coordinate
```

```
            XRight=s*cosd(Speaker2_azumith); %Right speaker coordinate
            YRight=s*sind(Speaker2_azumith); %Right speaker coordinate
            Theta=Phantom_desired_azumith
            [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta);
            g1=gCorrect (1);      %speaker 1 gain
            g2=gCorrect (2);      %speaker 2 gain
            g3=0;                 %speaker 3 gain
            g4=0;                 %speaker 4 gain
            L1z=s*sind(Speaker1_elavation);
            d=sqrt(s^2-L1z^2)
            L1x=d*cosd(Speaker1_azumith);
            L1y=d*sind(Speaker1_azumith);
            L2z=s*sind(Speaker2_elavation);
            d=sqrt(s^2-L2z^2)
            L2x=d*cosd(Speaker2_azumith);
            L2y=d*sind(Speaker2_azumith);
            L3z=s*sind(Speaker3_elavation);
            d=sqrt(s^2-L3z^2)
            L3x=d*cosd(Speaker3_azumith);
            L3y=d*sind(Speaker3_azumith);
            L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
            px=s*cosd(Phantom_desired_azumith);
            py=s*sind(Phantom_desired_azumith);
            pz=sqrt(px^2+py^2) *tand(Phantom_desired_elavation);
            p= [px py pz];
else
            %Do nothing
end

[Tone, FS, t] =Tone_Generator (400,1,2,0,0);
Y(:,1) =Tone*g1; %Channel 1
Y(:,2) =Tone*g2; %Channel 2
Y(:,3) =Tone*g3; %Channel 3
Y(:,4) =Tone*g4; %Channel 4
audiowrite ('3DStatic17.wav’, Y, FS);

figure
plot3([0 L1x], [0 L1y], [0 L1z],'r-o','linewidth',2)
hold on
plot3([0 L2x], [0 L2y], [0 L2z],'-o','linewidth',2)
plot3([0 L3x], [0 L3y], [0 L3z],'c-o','linewidth',2)
plot3([0 px], [0 py], [0 pz],'k-x','linewidth',2)
plot3([L1x L3x], [L1y L3y], [L1z L3z],'g','linewidth',2)
plot3([L2x L3x], [L2y L3y], [L2z L3z],'g','linewidth',2)
plot3([L1x L2x], [L1y L2y], [L1z L2z],'g','linewidth',2)
grid on
```

```
legend ('Left Speaker’, ‘Right Speaker’, ‘Bottom Speaker’, ‘Virtual Source’, ‘Active Triangle')
strl=sprintf ('Left Speaker Gain=%f’, g1);
text (-2, -2, -1.5, strl)
strl=sprintf ('Right Speaker Gain=%f’, g2);
text (-2, -2, -1.75, strl)
start=sprintf ('Bottom Speaker Gain=%f’, g4);
text (-2, -2, -2, strt)
xlabel ('Distance in Feet')
ylabel ('Distance in Feet')
label ('Distance in Feet')
title ('Virtual Source and Speaker Vectors Locations')
axis('square')
```

3D Dynamic Localization

```
close all;
clear all;
clc;

s=6;

%Speaker Locations in terms of degrees
Speaker1_azumith=30;
Speaker1_elavation=0;
Speaker2_azumith=-30;
Speaker2_elavation=0;
Speaker3_azumith=0;
Speaker3_elavation=30;
Speaker4_azumith=0;
Speaker4_elavation=-30;

Phantom_desired_azumith_start=-15;
Phantom_desired_azumith_end=15;
Phantom_desired_elavation_start=15;
Phantom_desired_elavation_end=-15;

Theta_azumith=linspace (Phantom_desired_azumith_start, Phantom_desired_azumith_end,100);
Theta_elavation=linspace(Phantom_desired_elavation_start,Phantom_desired_elavation_end,100
);

for i=1: length(Theta_azumith)
        if Theta_elavation(i)>0 & Theta_azumith(i)~=0 %Upper triangle
                L1z=s*sind(Speaker1_elavation);
                d=sqrt(s^2-L1z^2)
                L1x=d*cosd(Speaker1_azumith);
                L1y=d*sind(Speaker1_azumith);
                L2z=s*sind(Speaker2_elavation);
                d=sqrt(s^2-L2z^2)
                L2x=d*cosd(Speaker2_azumith);
                L2y=d*sind(Speaker2_azumith);
                L3z=s*sind(Speaker3_elavation);
                d=sqrt(s^2-L3z^2)
                L3x=d*cosd(Speaker3_azumith);
                L3y=d*sind(Speaker3_azumith);
                L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
                px=s*cosd(Theta_azumith(i));
                py=s*sind(Theta_azumith(i));
                pz=sqrt(px^2+py^2) *tand(Theta_elavation(i));
                p= [px py pz];
                g=p*inv(L)
                if max(g)>1
                        n=sqrt (g (1) ^2+g (2) ^2+g (3) ^2)
                        g=g/n
                end
        g1(i)=g (1);    %speaker 1 gain
        g2(i)=g (2);     %speaker 2 gain
        g3(i)=g (3);     %speaker 3 gain
        g4(i)=0;      %speaker 4 gain
        elseif Theta_elavation(i)<0 & Theta_azumith(i)~=0 %lower triangle
                L1z=s*sind(Speaker1_elavation);
                d=sqrt(s^2-L1z^2)
                L1x=d*cosd(Speaker1_azumith);
                L1y=d*sind(Speaker1_azumith);
                L2z=s*sind(Speaker2_elavation);
                d=sqrt(s^2-L2z^2)
                L2x=d*cosd(Speaker2_azumith);
                L2y=d*sind(Speaker2_azumith);
                L3z=s*sind(abs(Speaker4_elavation));
                d=sqrt(s^2-L3z^2)
                L3x=d*cosd(abs(Speaker4_azumith));
                L3y=d*sind(abs(Speaker4_azumith));
                L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
                px=s*cosd(Theta_azumith(i));
                py=s*sind(Theta_azumith(i));
                pz=sqrt(px^2+py^2) *tand(abs(Theta_elavation(i)));
                p= [px py pz];
                g=p*inv(L)
        if max(g)>1
                n=sqrt (g (1) ^2+g (2) ^2+g (3) ^2)
                g=g/n
        end
        g1(i)=g (1);     %speaker 1 gain
        g2(i)=g (2);     %speaker 2 gain
        g3(i)=0;       %speaker 3 gain
        g4(i)=g (3);     %speaker 4 gain
        % Make certain z negative for graphing purposes
        L3z=-s*sind(abs(Speaker4_elavation));
        pz=-sqrt(px^2+py^2) *tand(abs(Theta_elavation(i)));
        elseif Theta_elavation(i)==0 & Theta_azumith(i)~=0 %Midspeakers
                XLeft=s*cosd(Speaker1_azumith);  %Left speaker coordinate
                YLeft=s*sind(Speaker1_azumith);  %Left speaker coordinate
                XRight=s*cosd(Speaker2_azumith); %Right speaker coordinate
                YRight=s*sind(Speaker2_azumith); %Right speaker coordinate
                Theta=Theta_azumith(i)
                [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta);
```

```
          g1(i)=gCorrect (1);     %speaker 1 gain
          g2(i)=gCorrect(2);      %speaker 2 gain
          g3(i)=0;                %speaker 3 gain
          g4(i)=0;                %speaker 4 gain
          L1z=s*sind(Speaker1_elavation);
          d=sqrt(s^2-L1z^2)
          L1x=d*cosd(Speaker1_azumith);
          L1y=d*sind(Speaker1_azumith);
          L2z=s*sind(Speaker2_elavation);
          d=sqrt(s^2-L2z^2)
          L2x=d*cosd(Speaker2_azumith);
          L2y=d*sind(Speaker2_azumith);
          L3z=s*sind(Speaker3_elavation);
          d=sqrt(s^2-L3z^2)
          L3x=d*cosd(Speaker3_azumith);
          L3y=d*sind(Speaker3_azumith);
          L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
          px=s*cosd(Theta_azumith(i));
          py=s*sind(Theta_azumith(i));
          pz=sqrt(px^2+py^2) *tand(Theta_elavation(i));
          p= [px py pz];
  elseif Theta_elavation(i)~=0 & Theta_azumith(i)==0 %top and bottom speaker
          XLeft=s*cosd(Speaker3_elavation);  %Left speaker coordinate
          YLeft=s*sind(Speaker3_elavation);  %Left speaker coordinate
          XRight=s*cosd(Speaker4_elavation); %Right speaker coordinate
          YRight=s*sind(Speaker4_elavation); %Right speaker coordinate
          Theta=Theta_elavation(i)
          [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta);
          g1(i)=0;                %speaker 1 gain
          g2(i)=0;                %speaker 2 gain
          g3(i)=gCorrect(1);      %speaker 3 gain
          g4(i)=gCorrect(2);      %speaker 4 gain
          L1z=s*sind(Speaker1_elavation);
          d=sqrt(s^2-L1z^2)
          L1x=d*cosd(Speaker1_azumith);
          L1y=d*sind(Speaker1_azumith);
          L2z=s*sind(Speaker2_elavation);
          d =sqrt(s^2-L2z^2)
          L2x=d*cosd(Speaker2_azumith);
          L2y=d*sind(Speaker2_azumith);
          L3z=s*sind(Speaker3_elavation);
          d =sqrt(s^2-L3z^2)
          L3x=d*cosd(Speaker3_azumith);
          L3y=d*sind(Speaker3_azumith);
          L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
          px=s*cosd(Theta_azumith(i));
```

```
                    py=s*sind(Theta_azumith(i));
                    pz=sqrt(px^2+py^2) *tand(Theta_elavation(i));
                    p= [px py pz];
          elseif Theta_elavation(i)==0 & Theta_azumith(i)==0 % Midspeakers
                    XLeft=s*cosd(Speaker1_azumith);  %Left speaker coordinate
                    YLeft=s*sind(Speaker1_azumith);  %Left speaker coordinate
                    XRight=s*cosd(Speaker2_azumith); %Right speaker coordinate
                    YRight=s*sind(Speaker2_azumith); %Right speaker coordinate
                    Theta=Theta_azumith(i)
                    [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta);
                    g1(i)=gCorrect(1);      %speaker 1 gain
                    g2(i)=gCorrect(2);      %speaker 2 gain
                    g3(i)=0;                %speaker 3 gain
                    g4(i)=0;                %speaker 4 gain
                    L1z=s*sind(Speaker1_elavation);
                    d=sqrt(s^2-L1z^2)
                    L1x=d*cosd(Speaker1_azumith);
                    L1y=d*sind(Speaker1_azumith);
                    L2z=s*sind(Speaker2_elavation);
                    d =sqrt(s^2-L2z^2)
                    L2x=d*cosd(Speaker2_azumith);
                    L2y=d*sind(Speaker2_azumith);
                    L3z=s*sind(Speaker3_elavation);
                    d=sqrt(s^2-L3z^2)
                    L3x=d*cosd(Speaker3_azumith);
                    L3y=d*sind(Speaker3_azumith);
                    L= [L1x L1y L1z; L2x L2y L2z; L3x L3y L3z];
                    px=s*cosd(Theta_azumith(i));
                    py=s*sind(Theta_azumith(i));
                    pz=sqrt(px^2+py^2) *tand(Theta_elavation(i));
                    p= [px py pz];
          else
                    %Do nothing
          end
end
[Tone, FS, t] =Tone_Generator (400,1,2,0,0);
vq1=interp1(linspace (0, max(t), length(g1)), g1, t);
vq2=interp1(linspace (0, max(t), length(g2)), g2, t);
vq3=interp1(linspace (0, max(t), length(g3)), g3, t);
vq4=interp1(linspace (0, max(t), length(g4)), g4, t);
Y(:,2) =Tone. *vq1; %Channel 1
Y(:,1) =Tone. *vq2; %Channel 2
Y(:,4) =Tone. *vq3; %Channel 3
Y(:,3) =Tone. *vq4; %Channel 4
audiowrite ('3DDynamic8.wav', Y, FS);
```

Tone Generator

```
function [Tone, FS, t] = Tone_Generator (Frequency, Amplitude, Duration, Delay, endtime)

FS=44100;    % Samples per second (sampling Frequency)
ts=1/FS;     % Sampling interval
t=0:ts: Duration; % How long the Tone will last

Number_Of_Zeros_1=round(FS*Delay);      % Pads the beginning of the tone with 0's
Number_Of_Zeros_2=round(FS*endtime);    % Pads the end of the tone with 0's

Tone=Amplitude. *sin(2*pi*Frequency*t);
Tone= [zeros (1, Number_Of_Zeros_1) Tone zeros (1, Number_Of_Zeros_2)];

t=0:ts:(Delay+endtime+Duration);
```

Gain Calculator

```
% This code calculates all the gains
function [gCorrect]=NonEquaDistant2D (XLeft, YLeft, XRight, YRight, Theta)
Left= [XLeft YLeft];
Right= [XRight, YRight];
Listener= [0 0];
angle=Theta;

Center= [Listener (1) (Left (2) +Right (2))/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;
x=s*cosd(angle)+Listener (1);
yL=s*sind(angle)+Listener (2);
P= [x yL];
thetaLdeg = atand ((Left (2)-Listener (2))/ (Left (1)-Listener (1)));
thetaRdeg = atand ((Right (2)-Listener (2))/ (Right (1)-Listener (1)));
thetaPdeg=atand ((P (2)-Listener (2))/ (P (1)-Listener (1)));
L = [(Left-Center)' (Right-Center)']'; %switched left and right in this line
%to make it to where g1 is assigned to the "right speaker*
G = P*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

if gCorrect (1)>1
```

```
        gCorrect (2) =gCorrect (2)/gCorrect (1)
        gCorrect (1) =1
elseif G (2)>1
        gCorrect (1) =gCorrect (1)/gCorrect (2)
        gCorrect (2) =1
end
```