University of Cape Town

EEE3097S

Engineering Design: Electrical & Computer Engineering

First Progress Report
Cover Page

Group: 9
NJMLUN002 —— NKMMOR001 —— PRTCHA018

18 September 2023

Simulating Acoustic Triangulation Using Time Difference of Arrival Measurements

Prepared for:

Faculty of Engineering and Built Environment

Department of Electrical Engineering

University of Cape Town

Prepared by:

Group 9

Date:

18/09/2021

Keywords:

Acoustic triangulation, time difference of arrival, sound source tracking, generalized cross-correlation

Terms of Reference

This report was requested by the Electrical Engineering Department at the University of Cape Town, for the Engineering Design 3rd year course, on the 14th August 2023. This Terms of Reference outlines the scope and objectives for the preparation and submission of a comprehensive report on a simulation conducted to achieve sound source localization utilizing an array of microphones. This report is intended for engineering professionals and stakeholders.

In particular, the department requires:

1. A description of the simulation environment and tools used and explanation for chosen approach.
2. clear and separate sections for the subsystems to be simulated and the system as a whole, clearly describing the simulations and/or experiments run on the subsystems and the system as a whole.
3. Present the overall system architecture, including the raspberry pi zeros and microphones.
4. Describe how the distributed sensor network is structured, making sure to incorporate this structure in your simulation model.
5. Explain the algorithms or techniques used for time difference of arrival (TDOA) calculation.
6. Share the outcomes of the simulation experiments for the simulated subsystems and the system as a whole, clearly separating these results into their own individual sections.
7. Present the accuracy and precision achieved in locating the acoustic signal within the grid.
8. Recommend potential improvements or modifications to enhance the accuracy or efficiency of the system.
9. Summary of the key findings from the simulation stage.

Summary

This report concerns the simulation of an acoustic triangulation system requested by the Electrical Engineering Department for locating the position of an object emitting an audio signal through air using four microphones as sensors and two Raspberry Pi Zero (Pi) single-board computers as the micro-control unit (MCU).

Background and Motivation

Acoustic triangulation plays a significant role in a variety of applications across a spectrum of domains. Acoustic triangulation can be applied in security systems, virtual reality, sound design and audio engineering, as well as medical technology. This report assimilates the principles and methodologies characterising acoustic triangulation through a description of the algorithms that were implemented in locating a sound source based on the location of an array of microphones. The department aims to use the results from the simulation in refining the practical hardware implementation.

Objectives of Investigation and Methodology

The purpose of this report is to describe the simulation setup and methodology, present and analyse simulation results, assess the accuracy and reliability of sound source localisation techniques, as well as provide recommendations for practical implementation and improvements.

Results of Investigation

Time Delay Estimation Subsystem:

Our simulations demonstrated that the Time Delay Estimation Subsystem performs admirably in estimating time delays between microphone pairs. We utilized the GCC-PHAT algorithm to accurately compute these delays, even in the presence of mixed-frequency sound signals. The subsystem's performance was consistent, showcasing precise time delay estimations as the source's distance from the reference microphone increased. These results affirm the subsystem's robustness and effectiveness in capturing crucial time delay information.

Triangulation & Sound-Tracking Subsystem:

The Triangulation & Sound-Tracking Subsystem exhibited remarkable capabilities in localizing sound sources within a 3D space. Through Time Difference of Arrival (TDoA) measurements, we successfully determined the direction of arrival for sound sources. This method, which relies on angles adjacent to the longest sound paths between microphone pairs, consistently produced accurate sound source localization results. Moreover, our simulations highlighted the subsystem's potential for real-time sound source tracking, making it a valuable tool for applications requiring dynamic sound monitoring.

Conclusions

Our investigation into acoustic triangulation for sound source localization and tracking has yielded promising results. The Time Delay Estimation Subsystem showcased its ability to provide precise time delay estimations, important for the accurate determination of sound source locations. The Triangulation & Sound-Tracking Subsystem proved its effectiveness in localizing sound sources in a 3D space and tracking their movements with precision.

Recommendations

Based on the facts that were collated and the conclusions that were drawn about the acoustic triangulation design, the following recommendations to improve the properties of the system have been made:

- While our simulations provided valuable insights, it is recommended to conduct real-world testing to validate the system's performance under practical conditions. Consider implementing the system in controlled environments with actual sound sources to assess its accuracy and reliability.
- Extend the investigation to evaluate the system's performance in the presence of background noise and multiple sound sources. Implement noise reduction techniques to enhance the system's robustness in complex acoustic environments.
- Explore opportunities for optimizing the system's computational efficiency, particularly for real-time applications. This may involve hardware acceleration or algorithmic enhancements to reduce processing time
- Investigate the feasibility of integrating the acoustic triangulation system into specific applications, such as security systems, virtual reality environments, and sound design tools. Assess the compatibility and potential for customization to meet specific user requirements.
- Consider developing a user-friendly interface for configuring system parameters, visualizing sound source locations, and monitoring tracking results. User interface improvements can enhance the system's usability and accessibility.
- Implement data logging and analysis features to record and analyze sound source localization results over time. This can be valuable for performance assessment and trend analysis.
- Develop calibration procedures to ensure accurate alignment of microphones and reference points. Calibration is essential for maintaining the system's accuracy over extended periods.

Glossary

angle of arrival or AoA, refers to the angle between the horizontal axis of microphone pairs and the direction from which an acoustic wave arrives at a given microphone. 4

generalized cross correlation with phase transform is an algorithm used in signal processing to estimate the time delay or time difference of arrival between two signals received by separate sensors or microphones. 6

microphone signal refers to the audio signal detected by a microphone audio sensor. 13

Raspberry Pi Zero or Pi, or Pi Zero, is a 1 GHz single-core computer with a 512 MB. 4

source an object located in a 3D space which emits an acoustic wave. 10

time difference of arrival or TDoA, is a technique used in signal processing and localisation for computing the position of a of a sound source by measuring the difference in the time of arrival of a signal at multiple sensors or receivers. 1, 4

time of arrival or ToA, refers to the time t_i when an acoustive wave arrives at a microphone m_i. 4

Contents

List of Figures

1 Introduction

1.1 Scope and Limitations

This report aims to investigate the performance of an acoustic triangulation system for sound source localization and tracking in a three-dimensional (3D) space. The scope encompasses simulations and evaluations of the system's subsystems, namely the Time Delay Estimation Subsystem and the Triangulation & Sound-Tracking Subsystem.

1.2 Plan of Development

The report on the Acoustic Triangulation System flows through several key sections, each serving a distinct purpose. The introduction Provides an overview of the report's objectives and scope. Hardware design details the system.

2 High Level Design

2.1 Overall System Design

2.1.1 Signal Acquisition Subsystem Hardware Design

- **Sound Source:** An external sound source emits audio signals that need to be localized and tracked within a designated area.
- **Microphone Array:** These are the sensors responsible for capturing the audio signals. The microphones are distributed strategically within the area of interest to maximize coverage. The microphone signals are then processed to estimate the TDoA and subsequently triangulate the sound source's position.
- **Grid Map:** The grid map illustrates the xy-plane with respect to a reference microphone located at the origin of the Cartesian axis. The dimensions of the grid map is in meters.
- **Signal Filters:** A bandpass filter selects suitable signal frequencies to remove effects due to background noise.
- **Raspberry Pi Zeros:** Each Raspberry Pi Zero is associated with a pair of microphones. These devices serve as local processing units for microphones in their vicinity. They synchronize the data acquisition process, perform necessary signal processing, and communicate with the CPU to collectively estimate the sound source's position.

2.2 Structure of Distributed Sensor Network

The distributed sensor network in our system is structured to efficiently capture audio data from multiple locations within the environment. This network structure is designed to incorporate the following elements into our simulation model:

- **Microphone Placement:** The microphones are strategically placed throughout the environment to ensure optimal coverage and capture audio signals from different angles. The precise locations of the microphones are predefined and play a crucial role in sound source localization accuracy.
- **Communication Infrastructure:** The Raspberry Pi Zeros are connected to microphones, forming a network. This network allows each Raspberry Pi Zero to communicate with its associated microphones and relay the captured audio data. In your simulation, you should model the communication links between the RPis and microphones, ensuring that data can be transmitted and received.

- **Synchronization:** To facilitate accurate time-based measurements, the Raspberry Pi Zeros and microphones are synchronized. This synchronization ensures that time delay measurements TDoA are accurate, a critical factor in sound source localization.
- **Data Aggregation:** The audio data captured by each microphone is aggregated and transmitted to the respective Raspberry Pi Zero for processing. The network structure ensures efficient data transfer and minimal latency.
- **Sensor ID:** Each microphone in the network is assigned a unique identifier or sensor ID. This ID is used in the simulation model to associate audio data with specific sensors, allowing the system to track the source of incoming signals accurately .

3 Hardware Design

3.1 Signal Acquisition Subsystem Hardware Design

- **Sound Source:** An audio source, making use of speakers on a smartphone, emits a sound which travels toward the microphone array.
- **Microphone Array:** These are the sensors responsible for capturing the audio signals. The microphones are distributed strategically within the area of interest to maximize coverage. The microphone signals are then processed to estimate the TDoA and subsequently triangulate the sound source's position. The system is comprised of 4 Adafruit I2S MEMS microphone breakout boards.
- **Grid Map:** The grid map illustrates the xy-plane with respect to a reference microphone located at the origin of the Cartesian axis. The dimensions of the grid map is in meters.
- **Signal Filters:** A bandpass filter selects suitable signal frequencies to remove effects due to background noise. The filters are constructed using LM358 op-amps as well as capacitors and resistors with values corresponding to the desired cut-off frequencies in the audible frequency range.
- **Sampling Rate and Buffer Size:**Set to 44.1 kHz for accurate representation of audible frequencies. The sample rate significantly factors into the number of samples which must be considered for efficient data handling and system memory usage.

3.2 Pi Synchronization Subsystem Hardware Design

- **Raspberry Pi Zeros:** The system consists of two Raspberry Pi Zero single-board computers. Each Raspberry Pi Zero is associated with a pair of microphones. These devices serve as local processing units for microphones in their vicinity. They synchronize the data acquisition process, perform necessary signal processing, and communicate with the CPU to collectively estimate the sound source's position. The CPU coordinates the entire system. It collects data from microphones, computes TDoA, performs triangulation, and tracks the sound source's movement in real-time. The CPU is responsible for updating the sound source's position and presenting it through a user interface. Raspberry Pi Zeros have a 32-bit ARMv6Z architecture with a ARM1176JZF-S single-core.
- **Synchronized System Clock:** Each CPU core in the Raspberry Pi Zero boards has a maximum clock speed of 1 GHz. The clocks on each board need to be synchronized using a universal clock system so that time difference of arrival measurements can have an interpretable meaning. The Pi synchronization subsystem sets the date and time on both clocks simultaneously using epoch time to achieve minimum delay during transfer of

data between boards. The system uses a battery powered RTC module to keep track of the time while the CPU is sleeping.

- **Subsystem Interfacing:** The micro-control unit is required to interface with each subsystem as well as manage synchronization of operations on both Pi computers. The signal acquisition subsystem communicates with the Pis using the I2S protocol. The distance to the microphones is computed by the time delay estimation subsystem then transferred to functions computing the location in the triangulation subsystem. Triangulation subsystem results are used by the sound-tracking subsystem to continuously evaluate the location of the emitting sound source. Above CPU clock speed, interfacing and execution time are also constrained by the available space of 512 MB per Pi. The time and memory cost for inserting and removing data from the hard disk memory was taken into consideration.

3.3 Time Delay Estimation Subsystem Hardware Design

- **Microphone Array:** Microphones detect a signal and transmit the filtered signals to the CPU for further processing.
- **Raspberry Pi Zero:** Raspberry Pi Zeros CPUs process the time delay for the audio signal for each microphone. Utilizes GCC-Phat algorithm for sound source location calculations.

3.4 Triangulation Subsystem Hardware Design

- **Microphone Array:** Four microphones are placed strategically to detect audio signals whose time of arrival is used to determine the distance to the location. The location of the microphone labelled m_1 serves as the reference point for all computations of distances and angles.
- **Raspberry Pi Zero Subsystem Interface:** The CPU is used to compute the angle of arrival (AoA) for each microphone in a pair. The AoA is used to determine the distance to the acoustic emitter and ultimately, the coordinates in three dimensions.
- **Grid Map:** The grid map is drawn on an A1 sized sheet of paper with microphone m_1 located at the origin. Distances to microphones and the source are measured in meters.

3.5 Sound-Tracking Subsystem Hardware Design

- **RTC Module:** The RTC module is used to ensure that the system time is updated frequently and with millisecond precision to account for the speed of sound which implies that signals travel fast between the source and the microphone.
- **Raspberry Pi Zeros Subsystem Interface:** The location of the sound source computed by the triangulation subsystem is continually updated and transferred to the sound-tracking sound system to find the new location of the sound source as it moves.

3.6 User Interface Subsystem Hardware Design

- **Hardware Components:** Raspberry Pi Zero for user interface control and display; A1 paper with a grid; display interface.
- **Interface:** Displays real-time updates and user-configurable parameters.

4 Simulation Setup

4.1 Simulation Environment and Tools

4.1.1 Simulated Hardware Components

- The audio signal from speakers was simulated as a sinusoidal wave consisting of multiple frequencies in the range between 0 Hz and 20 kHz corresponding to audible frequency range.
- The Raspberry Pi Zero was emulated using Qemu (Quick Emulator) on a virtual machine simulating the ARMv6Z architecture and ARM1176JZF-S CPU. The ARM1176JZF-S is a reduced instruction-set processor offering subword parellism and digital signal processor-like operations allowing floating point multiplication. The emulated Pi had 256 MB.
- The Adafruit I2C MEMs microphones were simulated as audio emitters located at a point (x_s, y_s, z_s) in a 3D coordinate system whose origin corresponds to the location of microphone m_1. The other microphones are labelled incrementally in a clock-wise manner as m_2, m_3 and m_4.
- Microphones were assumed to have zero attenuation.

4.1.2 Software Components

- A full sound-tracking simulation was performed using MATLAB.
- Time delay estimations were simulated using the Python coding language. We also used Python to simulate triangulation of the sound source.

4.1.3 Tools and Libraries

- MATLAB code adapted from the MathWorks Object Tracking Using TDoA example.
- Time delay estimations were simulated using the Python coding language. We also used Python to simulate triangulation of the sound source.
- Python `numpy` library for mathematical functions used in computations of the AoA, TDoA and distances such as `numpy.arccos` for computing the angle subtended by two lines representing the path taken by the sound signal from the source to the microphone, and `numpy.array` to represent coordinates in the 3D axes.
- The Python `scipy` library was used to simulate properties of signal propagation to approximate the signals accurately. For example, the `scipy.signal.hilbert` function was used to compute the analytical signal using the Hilbert transform.
- The `scipy.signal.correlate` function was used to compute the correlation between the signals for each microphone pair. This function was implemented in the generalized cross correlation with phase transform algorithm to determine the time delay and time difference of arrival measurements.
- We used the Python `matplotlib` library to plot results for time delay simulations.

4.1.4 Functionalities

MATLAB Acoustic Triangulation System Simulation using TDoA Technique

The MATLAB simulation environment was designed to:

- Define a 3D scenario using the `computeCreateSingleTargetTDOAScenario` function which returns a an object representing a single sound source moving at a constant velocity.
- Detect by four stationary and spatially-separated microphones. The compute function also returns a matrix of two columns representing the pairs formed by the receivers.
- Simulate TDoA measurements from the objects using the `computeSimulateTDOA` function which allows the user to specify the measurement accuracy in the TDoA. The compute function specifies the measurement accuracy in the TDoA.
- Generate the TDoA detections from the source using the `computeTDOA2Pos` function to estimate the position and position covariance of the acoustic source.
- Calculate the emission time of an object given its position and obtained ToA detections using `computeCalcEmissionTime`.

Python Acoustic Triangulation System Simulation using TDoA Technique

The Python simulation environment was designed to:

- Allow the user to configure microphone positions, sound source location, and temperature.
- Set the microphone array configuration (default or user-defined).
- Set up the sound source location (default or user-defined).
- Set the ambient temperature to calculate the speed of sound in air (default at $25\,^\circ\,^\circ\mathrm{C}$ corresponding to a speed of sound of $343\,\mathrm{m\,s^{-1}}$).
- Generate sound signal.
- Simulate the propagation of the sound signal from the source to the microphone array.
- Calculate time delays for each microphone based on the source location and the speed of sound and apply time delays based on distances.
- Implement the generalized cross-correlation with phase transform algorithm to estimate the time delay between the reference microphone (m_1) and the other microphones.
- Use time delays to estimate angle to source relative to pairs of microphones using trigonometric calculations.
- Use estimated angles and the known microphone positions to perform triangulation of source in 3D coordinate system.
- Visualise the microphone array, sound source and estimated source location using a 3D plot.

4.2 TDoA Simulation Assumptions

4.2.1 Time Delay Estimation Simulation Assumptions

The TDoA positioning technique allows localization and tracking of a sound source by using the difference of the signal arrival times at sensors with different spatio-temporal properties. Suppose an acoustic signal is emitted at time t_e with propagation speed c. Then, in a signal acquisition array with two microphones m_1 and m_2, the time-of-arrival of the signal at each sensor located distances of r_1 and r_2 from the object, is given by

$$t_1 = t_e + \frac{r_1}{c} \tag{1}$$

$$t_2 = t_e + \frac{r_2}{c} \tag{2}$$

The signal arriving at each sensor is a time-shifted representation of the emitted signal from the source, at an unknown time t_e. TDoA exploits the difference between the time-of-arrival of the signal at each sensor to compute the source location. Given that the sound source s is located at the coordinate (x_e, y_e, z_e) in three-dimensional space, and that the sensors are located at (x_1, y_1, z_1) and (x_2, y_2, z_2), respectively, the time difference measurement t_1–t_2 corresponds to a difference in the distance between the sensors, i.e.

$$\sqrt{(x_t–x_1)^2 + (y_t–y_1)^2 + (z_t–z_1)^2} - \sqrt{(x_t–x_2)^2 + (y_t–y_2)^2 + (z_t–z_2)^2} = c \cdot (t_1–t_2) \tag{3}$$

The resulting graphical representation of the non-linear relationship in the above equation is a hyperbola with foci at the two microphone sensors, called constant TDoA curves [**?**]. This is illustrated in Figure **??**, assuming that c is the speed of sound through air taken to be $350\,\mathrm{m\,s^{-1}}$.

This method of calculating the TDoA measurement from the acoustic signal where microphones measure the ToA t_1 and t_2 and a difference in time-arrivals is determined, is useful in this application because the audio signal can be detected by the receiver from the leading edge.

The second method involves recording and transmitting the received signal to the CPU where a cross-correlation between received signals can be calculated. Here, the time difference of arrival is determined as the delay which maximizes the cross-correlation between two signals.

$$TDoA = \arg max[S_1 * S_2](t) \tag{4}$$

where $[S_1 * S_2](t)$ is the cross-correlation between the sound waves at microphones as a function of the time-delay t. The cross-correlation is evaluated from $-T_max$ to T_max, corresponding to the maximum delay give by

$$T_{max} = d/c \tag{5}$$

where d is the distance between the receivers. The synchronized clock of the Raspberry Pi Zeros fulfills the requirement for synchronization to achieve TDoA calculations for multiple microphones.

4.2.2 Triangulation Simulation Assumptions

We assume the use of a simplified triangulation technique to estimate the location of the sound source. This technique is based on the TDoA measurements calculated in the time delay estimation subsystem. For simplicity, we analyse the system using a two-dimensional approach where the microphone array is observed in the xy-plane and xz-plane. While the real system operates in three dimensions, reducing it to two dimensions allows us to visualize and understand the concept more easily.

We assume that the TDoA measurements obtained from the time delay estimation subsystem are ideal and contain no measurement errors. In practice, there may be inaccuracies due to noise and other factors. The positions of the microphones are assumed to be fixed and known in advance. The positions may change slightly over time or may not be precisely known, leading

to potential inaccuracies in the triangulation.

We assume perfect synchronization between the Raspberry Pi Zeros and the microphones. This synchronization is crucial for accurate triangulation, but in real-world scenarios, synchronization may not be perfect. The simulation assumes that the acoustic signals travel directly from the source to the microphones without any reflections or obstructions. In practice, reflections and obstacles can complicate the triangulation process. We focus on tracking a single sound source in the simulation. In real applications, there may be multiple sound sources, and the tracking algorithm would need to distinguish between them. We also assume a maximum operating range of 30 m.

4.2.3 Sound-Tracking Simulation Assumptions

4.3 Simulation Methodology

4.3.1 Time Delay Estimation Subsystem Algorithm Implementation

TDoA measurements were simulated from a single sound source and used to localize and track the source in the absence of noise and any other signals that can lead to missed detections. Adapted from the Mathworks "Object Tracking Using Time Difference of Arrival (TDOA)" simulation. A 3-D scenario was defined using a combination of `compute` functions, adapted from MathWorks `helper` functions. The `computeCreateSingleTargetTDOA` function returns a trackingScenarioRecording object representing a single sound source moving at a constant velocity. The simulated sound source was observed by 4 stationary, spatially-separated microphone sensors. The `computeCreateSingleTargetTDOA` function also returns a matrix of two columns representing the pairs TDOA pairs formed by the receivers. Each row represents the PlatformID of the platforms that form the TDoA measurement.

Using the `computeSimulateTDOA` function, we simulated the TDoA measurements from the objects. This `compute` function allowed for specifying the measurement accuracy in the TDoA. The timing inaccuracy in the clocks as well as TDoA estimations were assumed to be 100 ns. The emission speed and units of the reported time were specified using the `computeGetGlobalParameters` function. The simulation varied the speed of sound in air at temperatures ranging between $10\,^\circ$°C and $35\,^\circ$°C to observe the behaviour of the system for different signal propagation velocities. The `computeSimulateTDoA` function returns the TDoA detections as a cell `arrayof objectDetection` objects.

Each TDoA detection was represented using the `objectDetection` objects with instance variables describing the:

- Time: the time-of-arrival timestamp in seconds
- Measurement: TDoA in nanoseconds
- Measurement noise: the uncertainty in TDoA in nanoseconds
- Sensor index: a unique identifier per microphone pair
- Measurement parameters: a 2-element struct array with two fields describing the position of the reference microphone and the other microphone.

We used the `computeTDOA2Pos` function to estimate the location and covariance of the acoustic source at every time step using the spherical intersection algorithm which assumes two spheres with well-defined centers and radii. This function assumed that all TDoAs were measured with

respect to the reference microphone m_1 and returned the estimated position and uncertainty covariance as an `objectDetection` object.

The simulation also evaluated the behaviour of the system based on the path taken by the sound source. The tracking accuracy, in meters, of the microphone based on the path traversed by the source was recorded and compared to the results from different configurations of the microphone. The simulation was also implemented to assess the effect of placing the microphones further apart near the cut-off limit of half the wavelength of an acoustic signal.

4.3.2 Triangulation Subsystem Algorithm Implementation

We used the triangulation simulation to determine the position of the sound source of a wideband audio signal by implementing the generalized cross-correlation (GCC) algorithm. This differed from TDoA measurements which involves the use of time-of-arrival at the different microphones to determine the direction of an incoming sound wave. In the first step, the simulation evaluated the TDoA using GCC with phase transformation (GCC-PHAT) confined to the xy-plane and consisting of two pairs of microphones.

The second step involved computing the position of the source by triangulation, where straight lines are extended from each pair of microphones along the direction of arrival determined using TDoA measurements. For four microphones, the algorithm produced two directions of arrival, θ_1 and θ_2, corresponding to the angle from the positive x-axis.

The figure below illustrates a top view of the sound source and the pairs of microphone arrays. Assuming that the microphone pairs are located at the (0,0,0) and $(L, 0, 0)$ we determined the source is located at $(x, y, 0)$, we determined the coordinates of the source at $(x, y, 0)$ from the two directions of arrival, θ_1 and θ_2, using

$$
\begin{aligned}
& L = y \tan\theta_1 + y \tan\theta_2 \\
\implies \quad & y = \frac{L}{\tan\theta_1 + tan\theta_2} && (6) \\
& x = y \tan\theta_1 && (7)
\end{aligned}
$$

The simulation created the two microphone arrays using two uniform linear array (ULA) along the x-axis of the global coordinate system represented by grid lines on the A1 grid paper. The phase center of the reference microphone pair was assumed to be at (0, 0, 0) while the second microphone sensor pair was (0.10,0,0) m. The direction of the signal from the source located at a point (x, y ,0) was taken to be in the negative y-direction as illustrated in figure

We defined audio signals with different properties and determined the behaviour of the system for different configurations of the microphone array. The range of frequencies taken into consideration start from 100 Hz and go up to 15 kHz. The simulation assumed a maximum operating range of 15 m and a speed of sound of $350\,\mathrm{m\,s^{-1}}$. We used acoustic chirp signals with noise for analysis since this waveform can be approximated using a Dirac impulse signal. The acoustic chirp was collected by the two pairs of omni-directional microphones spaced at 0.1 m apart. The azimuth and elevation angles of the acoustic chirp were changed to determine the effectiveness of the configuration design.

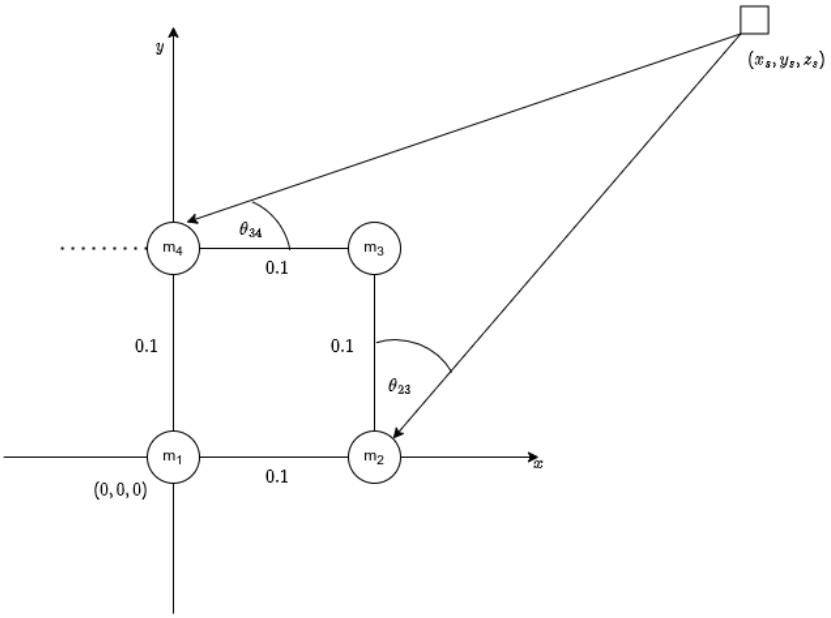

Figure 4.1: Illustrating the angles of arrival for microphone pairs where the angle θ_{ij}, for $i \neq j$, corresponds to the microphone pair m_i and m_j.

5 Simulation Results & Discussion

5.1 Time Delay Estimation Subsystem Simulation Results

Microphones were located at shown in the table below, where microphone m_1, located at the origin, was used as the reference microphone. The position of the source was changed to determine the accuracy of the time delay algorithm. The example below illustrates the setup of the simulation where the source was located at $(1, 1, 1)$ and the microphones were positioned in a square configuration with each side having a length of 0.1 m. This was chosen to satisfy the condition that the distance between microphone pairs cannot be less than half the wavelength of sound waves.

The audio signal (`sound_signal` variable in 1) was simulated as a mixed sinusoid as illustrated by the code in listing 1.

Listing 1: Showing code implementation of an audio signal with mixed freqencies ranging between 440 Hz and 1760 Hz.

```
# Generate a sound signal (sine wave)
sample_rate = 44100  # Sample rate in Hz
duration = 0.05  # Duration in seconds
num_samples = int(duration * sample_rate)
frequencies = [440.0, 880.0, 1320.0, 1760.0]  # Frequencies in Hz
t = np.linspace(0, duration, int(sample_rate * duration), endpoint=False

amplitudes = [0.8, 0.6, 0.4, 0.2]  # Corresponding amplitudes

# Generate the mixed signal
mixed_signal = np.zeros(num_samples)
for freq, amp in zip(frequencies, amplitudes):
mixed_signal += amp * np.sin(2 * np.pi * freq * t)
# Generate the sound signal (sine wave)
sound_signal = mixed_signal
```

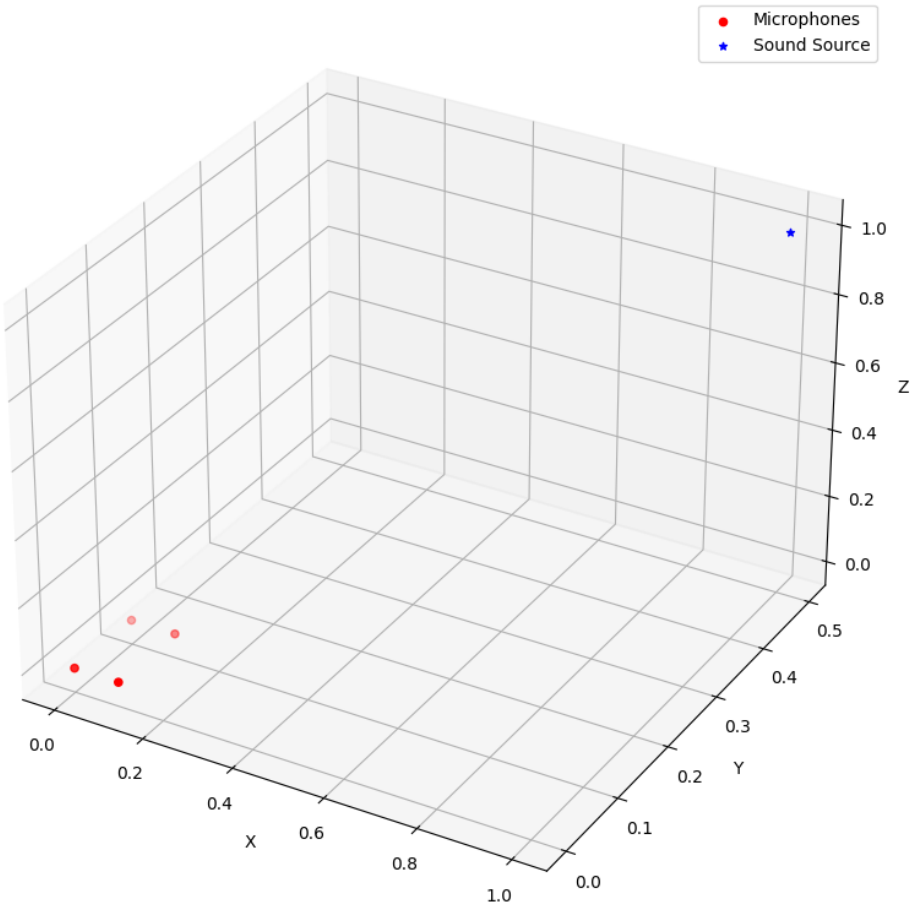

Figure 5.1: Showing a 3D image of the simulation setup where microphone m_1 was located at $(0, 0, 0)$ and the source was located at $(1, 1, 1)$. The microphones were spaced by 0.1 m apart corresponding to less than half the wave length of sounds in the audible frequency range.

The propagation of sound through air was simulated using the `simulate_sound_propagation` function which had three arguments as shown in 2. The default settings of the simulation used a speed of sound of 346 m s corresponding to an ambient temperature of 25°.

Listing 2: Showing sound propagation simulated given a sinusoidal sound wave, the source position, and microphone positions in 3D.

```
# Sound propagation simulation
def simulate_sound_propagation(sound_signal, source_position,
microphone_position, speed_of_sound, sample_rate):
# Calculate the distance from the source to the microphone
distance = np.linalg.norm(source_position - microphone_position)
# Calculate the time it takes for sound to travel
#from the source to the microphone
travel_time = distance / speed_of_sound
# Calculate the corresponding sample delay
sample_delay = int(travel_time * sample_rate)

# Create a delayed version of the sound signal
delayed_sound_signal = np.zeros_like(sound_signal)
```

```
    if sample_delay < len(sound_signal):
    delayed_sound_signal[sample_delay:] = sound_signal[:-sample_delay]

    return delayed_sound_signal

    microphone_signals = []
    for mic_position in microphone_positions:
    microphone_signal = simulate_sound_propagation(sound_signal, source_posit
    microphone_signals.append(microphone_signal)
```

The signal at each microphone after implementing the `simulate_sound_propagation` function is shown in figure 5.2 for a duration of 0.5 s. The figure shows that the signals at each microphone are identical although slightly shifted in time to simulate time delays. Figure 5.3 shows the

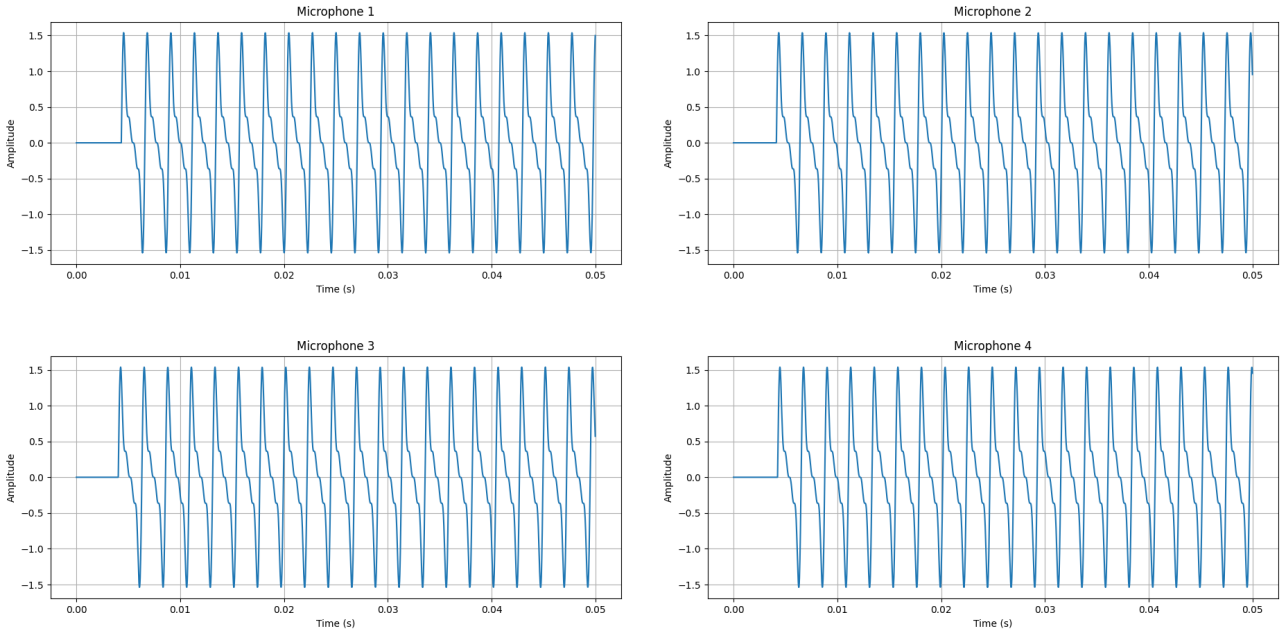

Figure 5.2: Showing audio signals at each microphone after simulating sound propagation through air using Python.

signals at the microphones when the source was moved further away to the point $(2, 2, 2)$. A comparison of the figures of the signals for sources located at different points showed accurate time-shift in the signals at the microphones as illustrated by a longer dead-time in the signals where the source was further away compared to a shorter dead-time for a closer source position. This was expected since the speed of sound was taken to be constant, meaning that a sound wave takes longer to reach the microphones when the source is further from the microphones. Time delay estimations were simulated using the code shown in 3 where the fast Fourier transform function was implemented to find the cross-correlation between the signals at the microphones.

Listing 3: Showing implementation of GCC-PHAT algorithm to estimate time delays.

```
# Compute the GCC-PHAT cross-correlation
def gcc_phat(signal1, signal2):
fft_signal1 = np.fft.fft(signal1)
fft_signal2 = np.fft.fft(signal2)
cross_correlation = fft_signal1 * np.conj(fft_signal2)
cross_correlation /= np.abs(cross_correlation)
```

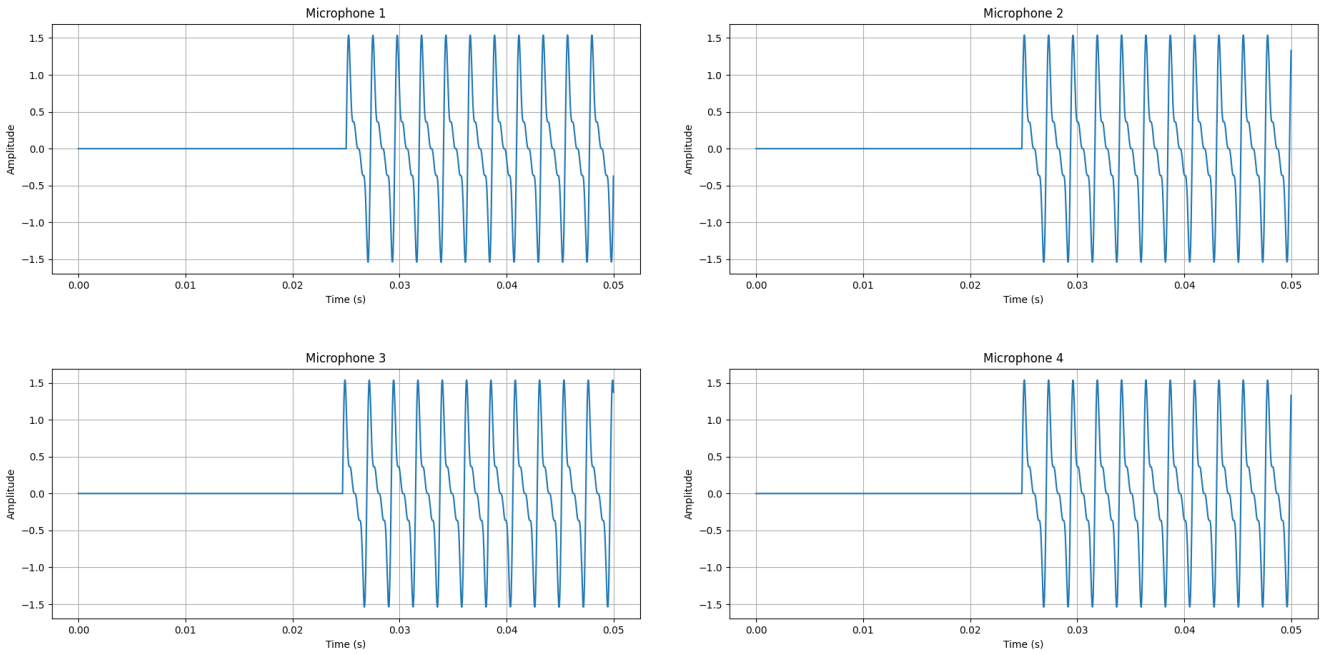

Figure 5.3: Showing a longer dead-time as sound takes longer to travel to microphones compared to the result shown in figure 5.2 above.

```
inverse_correlation = np.fft.ifft(cross_correlation)
gcc_phat_result = np.abs(inverse_correlation)
return gcc_phat_result
```

Results from the implementation of the GCC-PHAT algorithm are illustrated in the figures below. The results show the GCC-PHAT value increased as the source moved away from the reference microphone, and the number of peaks, corresponding to the number of frequencies in the mixed acoustic signal, also increased. This is illustrated by figures 5.4 and 5.5 showing the GCC-PHAT values obtained when the source was located at points $(1, 1, 1)$, $(2, 2, 2)$, and $(5, 5, 5)$.

GCC-PHAT simulations also showed that the correlation between microphone signal signals decreased as the source moved further away from the source as illustrated by the plots below. This result corresponds to the expected behaviour where information about signals is attenuated as it propagates through a medium for a longer distance.

5.2 Triangulation & Sound-Tracking Subsystem Simulation Results

Triangulation was simulated using MATLAB where TDoA measurements from a single sound source were used to to localize and track the source in the absence of noise and any other signals that could lead to missed detections. In the triangulation algorithm, time delay estimations were used to determine the direction of arrival of the source as shown in figure 5.9. Using the TDoA measurements, the angle of arrival was found to be always adjacent to the longest path taken by sound to reach a microphone pair

The MATLAB scrip code snippet shown in listing 4 tracking shows an implementation of the `computeSimulateTDOA` function used to simulate the TDoA arrival using the first method described above where the time of arrivals are used to compute the time delay estimations.

Listing 4: Showing while loop for continuously tracking the microphone location.

```
% Display object
display = ComputeTDOATrackingExampleDisplay(XLimits=[-1e4 1e4],...
```

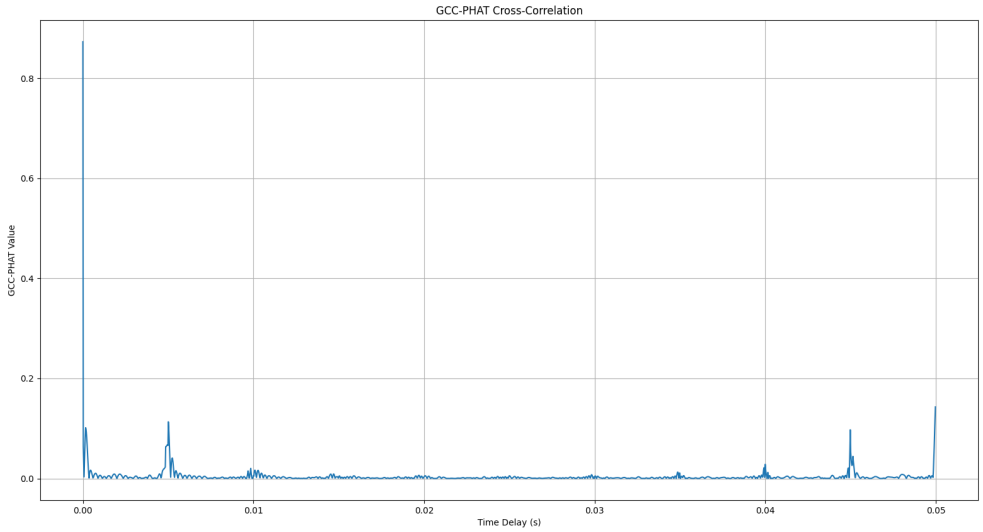

Figure 5.4: Showing the GCC-PHAT value vs time plot for the cross-correlation analysis between two signals received by microphone pairs when the source was a meter away from the reference microphone.

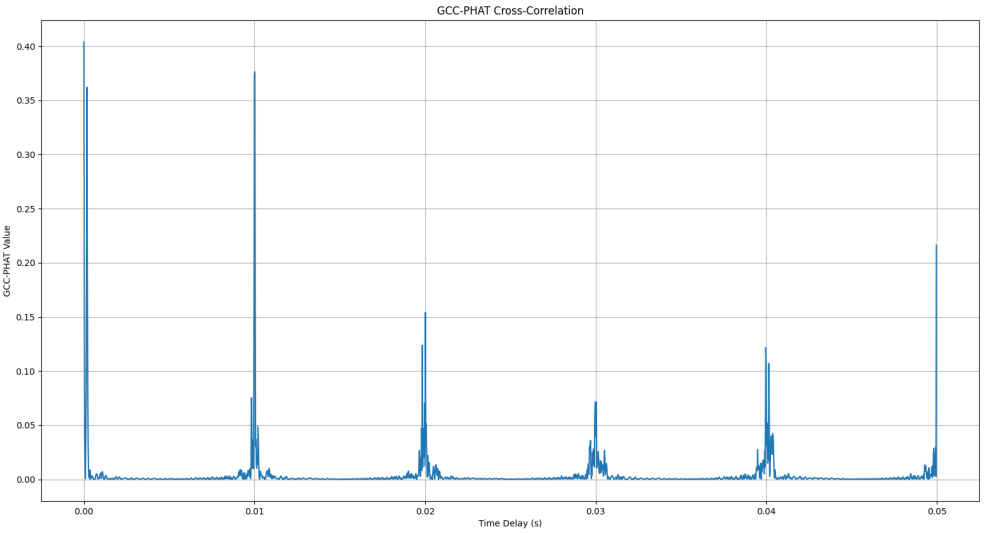

Figure 5.5: Showing the GCC-PHAT value vs time plot for the cross-correlation analysis between two signals received by microphone pairs when the source was 2 meters away from the reference microphone. More peaks were prevalent as the distance from the origin to the sound source increases.

```
YLimits=[-1e4 1e4],...
LogAccuracy=true,...
Title="Single Object Tracking");

% Create a GNN tracker
tracker = trackerGNN(FilterInitializationFcn=@computeInitHighSpeedKF,...
```

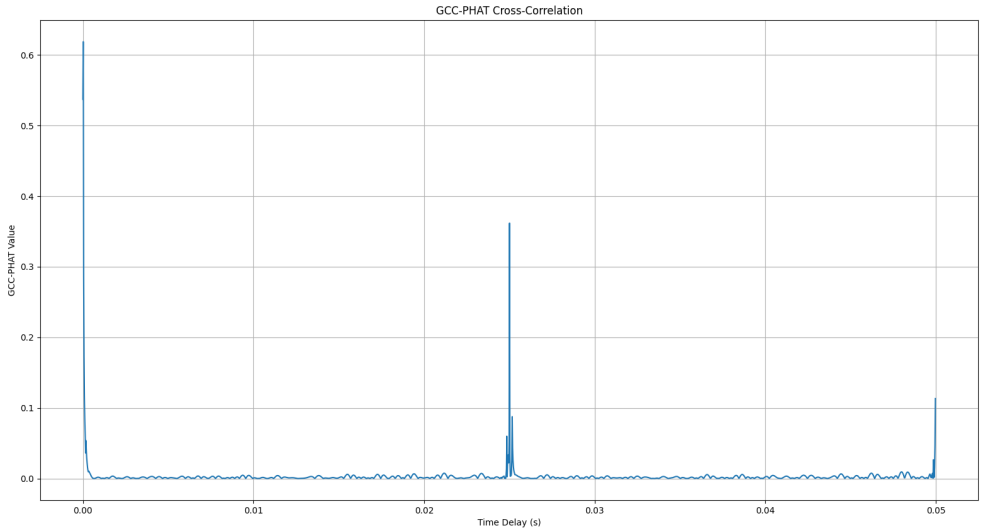

Figure 5.6: Showing the larger GCC-PHAT value at zero for sound source located further away.

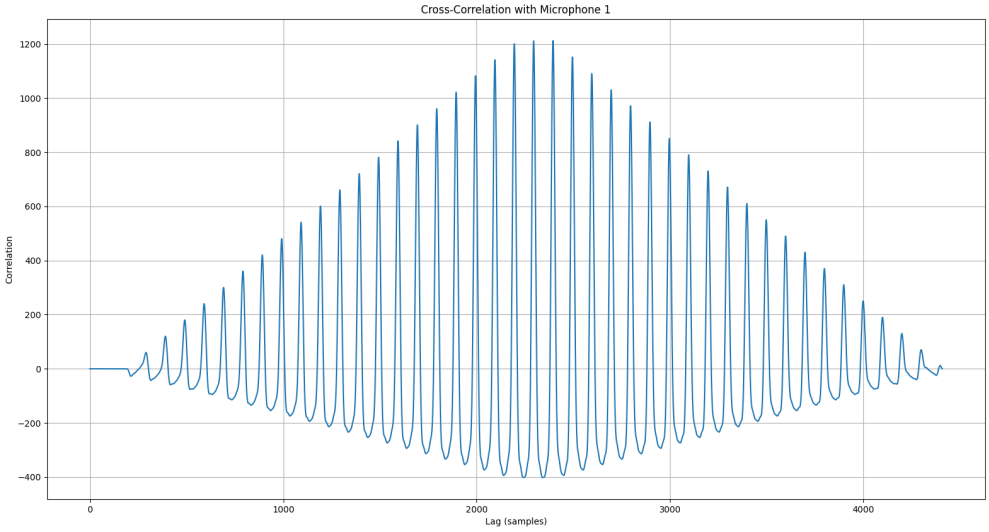

Figure 5.7: Cross-correlation plot when the source was located at point (1,1,1). The correlation between the signals is highest when the sound emitter is closer to the origin.

```
AssignmentThreshold=100);

while advance(scenario)
% Elapsed time
time = scenario.SimulationTime;

% Simulate TDOA detections without false alarms and missed detections
tdoaDets = computeSimulateTDOA(scenario, rxPairs, measNoise);
tdoaseconds = tdoaDets(2)
% Get estimated position and position covariance as objectDetection
```

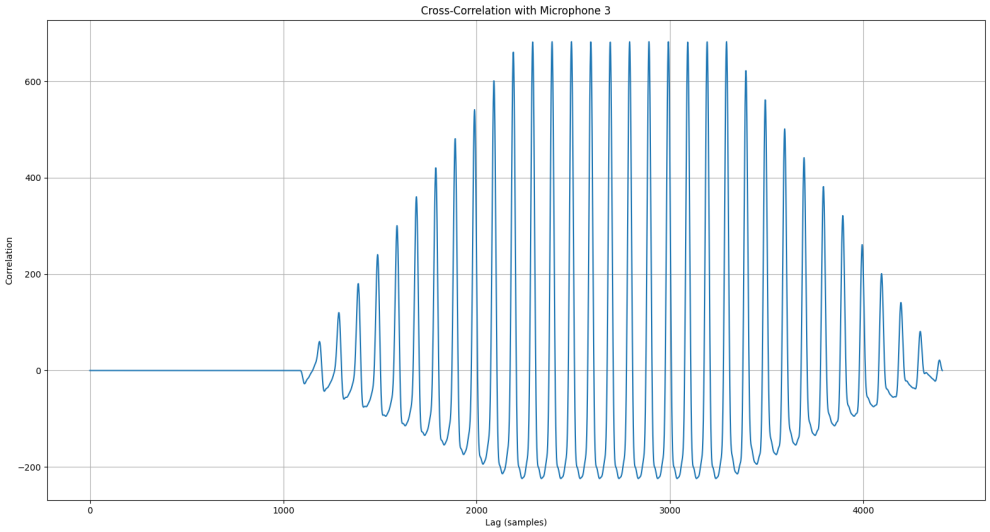

Figure 5.8: Cross-correlation between microphone signals decreases as the distance to the microphone increases. Here, the sound source was located at point (5,5,5).

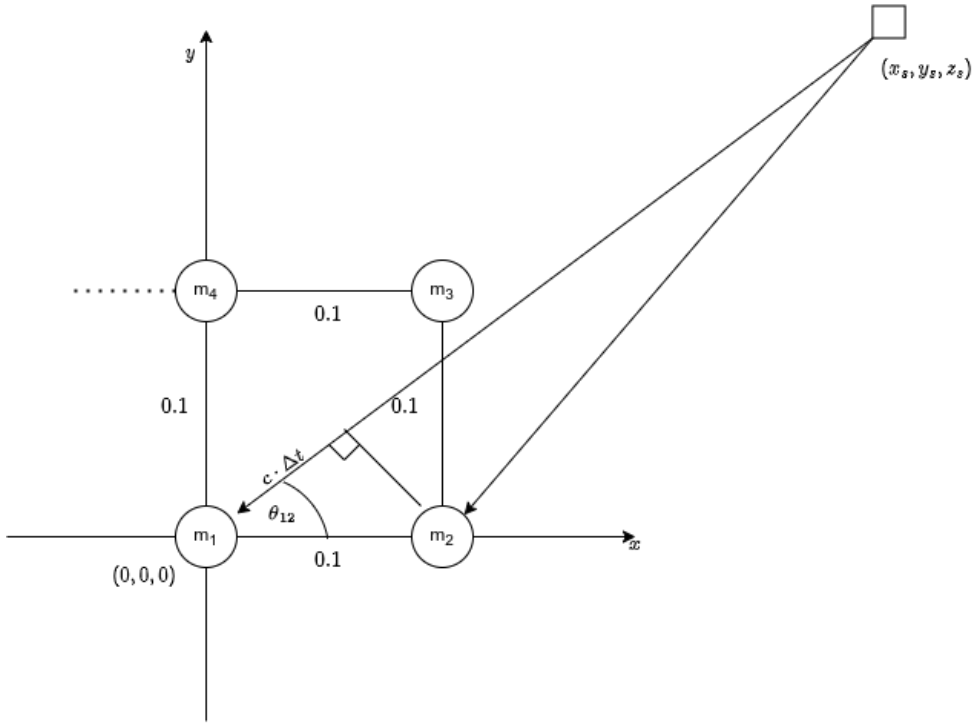

Figure 5.9: Angle of arrival used to compute the direction of arrival for microphone pairs.

```
% objects
posDet = computeTDOA2Pos(tdoaDets,true);

% Update the tracker with position detections
tracks = tracker(posDet, time);

% Display results
```

```
display(scenario, rxPairs, tdoaDets, {posDet}, tracks);
end
```

The simulation used the `computeTDOA2Pos` function to estimate the position and position covariance of the acoustic source at every step using the spherical intersection algorithm. Results were illustrated using an animation as shown in figure 5.10 where a sound source moves along a track and emits sound towards the microphone array.

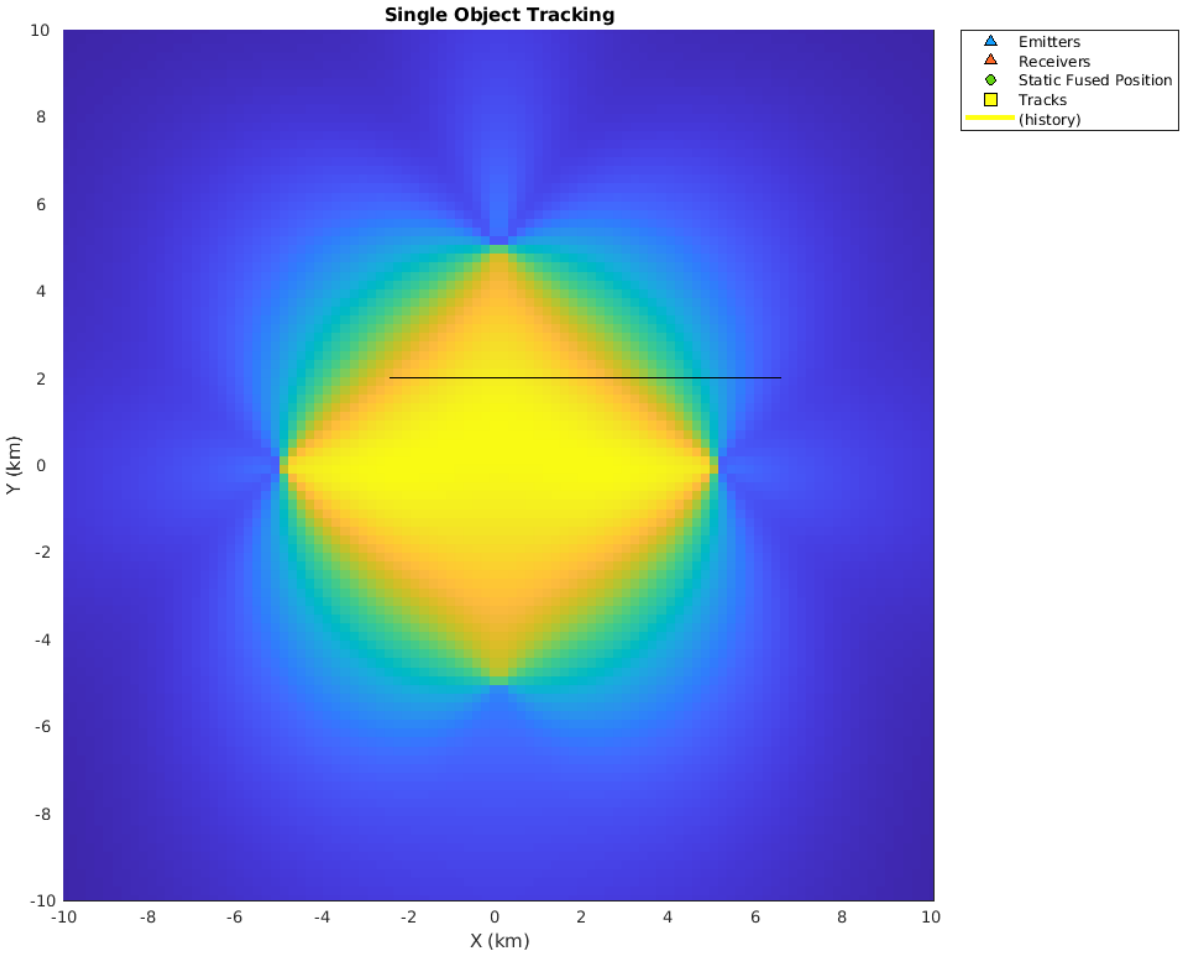

Figure 5.10: Showing sound-tracking simulation in MATLAB where a sound source moves at a constant speed along a straight track.

6 Acceptance Test Procedures

6.1 Defining Acceptance Test Procedures

Localization Accuracy:
- Deviation between estimated and actual sound source positions. This will provide insight into the system's precision in determining sound source locations within the grid

Real-time Performance:
- Processing time for sound source localization. This assesses the system's efficiency in processing data in real-time scenarios.

Noise Robustness:
- Ability to accurately localize sound sources in noisy environments. The Signal-to-Noise Ratio (SNR) will be used to quantify the noise level at which accurate localization can still be achieved.

Computational Efficiency:
- Resource utilization and execution time of algorithms.

System Reliability:
- Consistency and stability of results across multiple tests

6.2 Outline of Transition to Physical Implementation

1. Hardware Procurement and Setup:

- Identify and procure the necessary hardware components, including Raspberry Pi Zero boards, MEMS microphones.

- Determine the physical locations for sensor deployment.

Install microphones at predefined positions, ensuring the same configuration as in the simulation.

2. Software Implementation:
2.1. Porting Simulation Code:

- Adapt simulation code to work on the Raspberry Pi Zero devices.
- Ensure that synchronization mechanisms are effectively implemented.

2.2. Real-time Processing Optimization:

- Enhance code for real-time performance.
- Optimize algorithms for efficient execution on the hardware.

2.3. Integration with Hardware:

- Establish communication protocols between Raspberry Pi Zero devices and microphones.

- Ensure data acquisition and transmission are functioning correctly.

3. Testing and Calibration:

3.1. Hardware Testing:

- Conduct hardware testing to ensure all components are functioning as expected.
3.2. Calibration Procedures:

- Implement calibration routines for microphone sensitivity and sensor positions.

4. Validation and Performance Testing:

4.1. Localization Accuracy Testing:

- Place sound sources at known positions within the physical environment.
- Record and compare estimated positions with actual positions.
- Validate the system's accuracy by calculating error metrics.

4.2. Real-time Performance Assessment:

- Measure the time it takes for the physical system to calculate sound source positions.

4.3. Noise Robustness Testing:

- Introduce controlled levels of background noise.
- Evaluate the system's ability to localize sound sources in noisy environments.

4.4. Computational Efficiency Analysis:

- Monitor CPU and memory usage during real-time processing.
- Compare execution times with simulation results.

5. Fine-tuning and Optimization:

5.1. Algorithm Optimization:

- Fine-tune algorithms based on physical system performance.
- Address any discrepancies between simulation and real-world behaviour.

6. Documentation and Reporting:

6.1. System Documentation:

- Document the physical system setup, hardware specifications, and software configurations.

6.2. Performance Report:

- Generate a comprehensive report detailing the physical system's performance.
- Include results of accuracy, real-time performance, noise robustness, and efficiency tests.

7. Deployment and Integration:

7.1. Integration with Application:

- Integrate the sound source localization system with the intended platform.

7 Conclusion

In this study, we conducted simulations to assess the performance of the Time Delay Estimation and Triangulation subsystems for sound source localization in a 3D space. The following key findings and conclusions can be drawn from our simulations:

7.1 Time Delay Estimation Subsystem

1. The Time Delay Estimation Subsystem demonstrated accurate estimation of time delays between microphone pairs, even for complex mixed-frequency sound signals. The GCC-PHAT algorithm successfully identified time delays, which increased as the source moved further away from the reference microphone.

2. The correlation between microphone signals decreased as the source moved farther from the reference microphone, reflecting the expected attenuation of signals over longer distances.

3. Simulation results validated the capability of the subsystem to provide precise time delay estimations, crucial for accurate sound source localization.

7.2 Triangulation & Sound-Tracking Subsystem

1. Triangulation simulations successfully localized sound sources in 3D space using Time Difference of Arrival (TDoA) measurements. The direction of arrival was computed using the angle adjacent to the longest sound path between microphone pairs.

2. The subsystem demonstrated its ability to accurately track sound sources' movements in a noise-free environment, making it suitable for applications requiring real-time sound tracking.

In conclusion, the simulations showcased the effectiveness and accuracy of the Time Delay Estimation and Triangulation subsystems for sound source localization and tracking in 3D space. Future work may involve incorporating noise models and real-world conditions to further validate the subsystems' performance in practical applications.

References

[1] MathWorks, "Object tracking using time difference of arrival (tdoa)." Available at `https://www.mathworks.com/help/fusion/ug/object-tracking-using-time-difference-of-arrival.html` (2023/09/19).

Appendix A: Link To Acoustic-Triangulation-Github Link

https://github.com/BongaNjamela001/Acoustic-Triangulation-System