# Acoustic Source Localization

ECE4873 Senior Design Project

Team Name – Spot
Project Faculty Advisor – Dr. Ma Xiaoli

Tiffany Ho – Major: EE, GT Account: tho65, email: tiffanyho@gatech.edu
Ajeetpal Dhillon – Major: EE, GT Account: adhillon30, email: adhillon30@gatech.edu,
Harry Nguyen – Major: CmpE, GT Account: hnguyen363, email: hnguyen363@gatech.edu
Sidong Guo – Major: EE, GT Account: sguo93, email: sguo93@gatech.edu
Andrew Dulaney – Major: EE, GT Account: adulaney, email: adulaney7@gatech.edu
Daniel Scarborough – Major: EE, GT Account: dscarborough3, email: dscarborough3@gatech.edu

Submitted

Submission Date February 11th

# Table of Contents

**Executive Summary (5 pts)** .......... ii

1. **Introduction (10 pts)** .......... 1

    1.1 Objective .......... xx
    1.2 Motivation .......... xx
    1.3 Background .......... xx

**2. Project Description and Goals (15 pts)** .......... xx

**3. Technical Specification (15 pts)** .......... xx

**4. Design Approach and Details (20 pts)**

    4.1 Design Approach .......... xx
    4.2 Codes and Standards .......... xx
    4.3 Constraints, Alternatives, and Tradeoffs .......... xx

**5. Project Demonstration (5 pts)** .......... xx

**6. Schedule, Tasks, and Milestones (10 pts)** .......... xx

**7. Marketing and Cost Analysis (5 pts)** .......... xx

    7.1 Marketing Analysis .......... xx
    7.2 Cost Analysis .......... xx

**8. Current Status (5 pts)** .......... xx

**9. Leadership Roles (5 pts)** .......... xx

**10. References (5 pts)** .......... xx

**Appendix A** .......... xx

**Appendix B** .......... xx

# Executive Summary

RF and Wi-Fi localization has been the center of research and development for many decades, but recent decades saw an increasing interest in acoustic localization with algorithms like ODAS (Open Embedded Acoustic System). This project explores the concept of source localization for acoustic signals. The objective of this project is to evaluate, research and devise an inexpensive, accurate algorithm for acoustic localization that incorporates tracking and can have estimation results displayed on a flexible GUI. Upon finishing the design, the software can be applied in classroom setting for automatic camera angle adjustment, surveillance camera and other sound controlled related applications. The project attempts to address the high complexity of other algorithms by utilizing a novel, less computationally expensive localization approach that utilizes both TDOA (Time Difference of Arrival) and RSS (Receive Signal Strength) to feed angle and distance information to the serial, which will then be displayed on a GUI. The angle information is established through average multiple microphones pair through far-field and an outlier filter whereas the distance information is obtained by constructing an indoor pathloss model. The core product of the project is software that does not cost anything and has parameter definition that is changeable for any microphone array positioning. To demonstrate and evaluate the project, we will use Raspberry Pi3 and its MATRIX VOICE module, which costs around 200 dollars, depending on the exact model and manufacturer. There are no other sources of expenditure. The project aims to achieve at least half meter accuracy in location estimation in a classroom setting and an operation range of at least 15 meters. The algorithm should also support sampling frequency no greater than industry standard 96 kHz and real time display on GUI. For future improvements, the algorithm can incorporate RF signal estimation and neural network for more sophisticated tracking algorithms. If time allows, the project will incorporate multi-source tracking and 3-D localization.

**Index:**

1. **TDOA (Time Difference of Arrival): A location estimation algorithm that uses the difference in arrival time of receive signal to extract position information.**

2. **RSS (Receive Signal Strength): A location estimation algorithm that uses the power of the receiving signal to deduce relative distance.**

3. **GCC (Generalized Cross-Correlation): A time-delay estimation method that uses cross-correlation with certain weighting to estimate the time difference between two signals.**

4. **PHAT (Phase Transform): A GCC weighting that helps to heighten the peak.**

5. **MATRIX: A microphone array module compatible with Raspberry Pi.**

# Acoustic Source Localization

## 1. Introduction

Acoustic source Localization using microphone array has been a researched topic ever since last century. With the development of communication systems, outdoor localization is dominated by GPS (Global Positioning System) and left little room for acoustic signals. As the focus of research shifts from outdoor to indoor, there is a rekindle of interests in acoustic signals. TDOA acoustic source localization is composed of two main steps: Time difference estimation and triangulation. The use of cross-correlation for estimation time delay between pairs of microphones was well documented in C. Knapp and G. Carter's paper. Throughout this century, different geometries of microphone and time estimation methods were continuously developed. The biggest challenge, however, remains the questions of obtaining location information from pairs of time estimates through triangulation or multilateration. The errors in time different estimation meant that a closed-form calculation of position is not possible and therefore requires an addition of an error term. New methods like Gillette-Silverman algorithm and other closed-form algorithms were developed and proposed in papers to meet this restriction. But after extensive testing and simulation, we decided that they were either inaccurate or very geometry specific and therefore lack versatility. The other method proposed uses far-field simplification, which in an indoor environment is unattainable and will always produce an overestimation. This project, however, will use multiple pairs of trigonometry calculation by assuming far-field, some of which will attain underestimation. Therefore, by averaging over multiple angle calculations, a reasonable value can be attained between upper and lower bounds. By far the most established algorithm is ODAS, which utilizes beamforming and Hierarchy Search Matching and as a result, though accurate, is extremely hard to translate once the geometry is changed.

### 1.1 Objective

Our objective is to create software that can detect the location of a sound that is generated in an area with desirable accuracy and simplicity. The software will also track the incoming sound and the positions will be reflected on the GUI. The algorithm will compensate for the inaccuracy and restrictiveness of some other models.

### 1.2 Motivation

While a lot of detection has been done using camera, infrared technology and Wi-Fi, our group would like to design software that can determine location based on incoming acoustic signals. We hope this software will be more accessible than infrared and more flexible than cameras. We would like owners to be able to place the device down in a room and be able to determine where a noise is coming from. This could be used in classroom settings or places that care for elderly people who might wander off, or for hyperactive kids.

### 1.3 Background

While there are multiple ways that a location can be determined, our project will focus on three. The first is time of arrival. TOA calculates the position of an acoustic signal based on the time it takes to reach a receiver. It is easy to implement, but it is not very practical because the exact timing from the noise source is usually not provided, we therefore decide to use this as a backup plan. Because the timing information is unknown in real world applications, our device will focus on a more refined method called time difference of arrival. TDOA can determine the direction of a source without knowing its starting time. It uses similar calculations to the time of arrival method, but it uses several receivers to capture the signal. Because sounds do not move instantaneously, we use the difference between the arrivals to determine the location of the source. This is a good method, but it heavily depends on the location of the project's microphones and estimation precision. Because we are using microphones that are space closely together, the distance cannot be calculated reliably. Therefore, we will use the receive signal strength method to determine the distance of the source. This method is not perfect because the original amplitude of the signal is not known, but it will be much more reliable than TDOA. Combining two methods will provide both angle and distance information the localization needs.

## 2. Project Description, Customer Requirements, and Goals

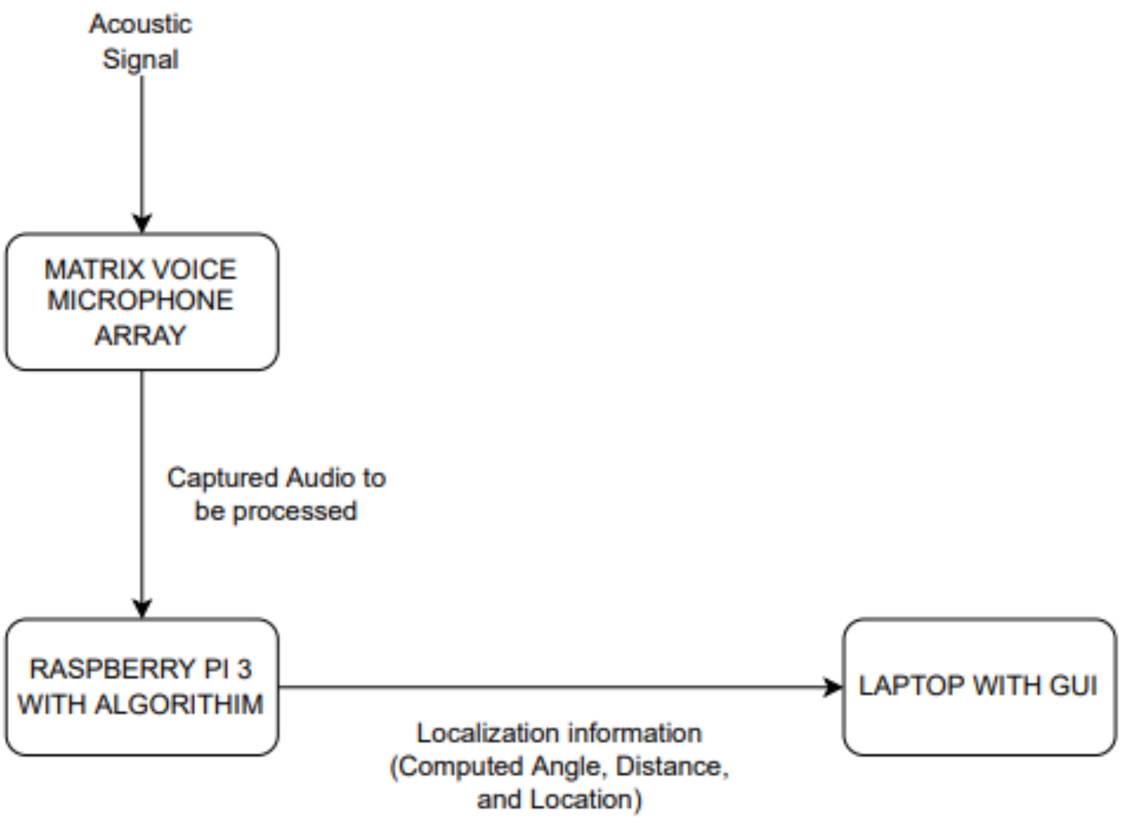

Acoustic Signal
MATRIX VOICE MICROPHONE ARRAY
Captured Audio to be processed
RASPBERRY PI 3 WITH ALGORITHIM
Localization information (Computed Angle, Distance, and Location)
LAPTOP WITH GUI

Figure 1. System Block Diagram

Our team aims to design a real time localization software that uses a Matrix Voice microphone array to capture acoustic signals. This design will also include a GUI to display the location, distance, and direction of the source relative to our device. This GUI will be viewable on an external computer or laptop. This design is intended to be accessible to all users who need to track acoustic or speech signals within a limited radius. To ensure this ease of accessibility, the goal is to make the UI easy to read and adjust for different circumstances. The goal is to create an algorithm with adjustable parameters to conduct point source localization in varying environments. This product could be used in a variety of fields: emergency response, surveillance systems, and speech/target tracking.

Our metric of evaluation in terms of customer needs are Accuracy, Convenience, Durability, Responsiveness, Flexibility and Cost. Each metric is quantified in the next section.

The project has several hardware and software constraints:

1. The software we designed interfaced mainly with Raspberry PI and its companion MATRIX module, however, the microphone parameters can be redefined and support other microprocessors if they are in C++.

2. The software does not support sampling rates higher than 96 kHz, in other words, the software is unresponsive to signals with frequency higher than 48 kHz, which is usually outside the range of speech signal.
3. Microphones need to be placed precisely with relative position to get an accurate localization.
4. The tracking algorithm does not regard two signals as one source after a certain time interval during which the signal is silent, the exact threshold will be explored during implementation.

## 3. Technical Specifications

Table 1. Design Parameters for the project

| Parameters | Specifications |
|---|---|
| Audio Specification | 8-96 kHz |
| Bit Depth | Signed 16 bits |
| Operation Range | 20 meters |
| Cost | Below $200 |
| Angle Accuracy | Average difference within 3 degrees |
| Distance Accuracy | Average difference within 30 centimeters |
| Processing Delay | Average processing delay within half second |

Table 2. House of Quality for the project

| Customer and Engineering Requirements | Customer importance (Most Important =5) | Design Criterion A | Design Criterion B | Design Criterion C | Design Criterion D |
|---|---|---|---|---|---|
| Easy to Use | 4 | X | Y | Z | W |
| Reliability | 5 | Y | Y | X | W |
| Durability | 3 | W | X | W | W |
| Low Price | 5 | W | W | W | X |
| Responsiveness | 2 | W | Y | Z | W |
| Flexibility | 3 | W | X | Z | W |

X: Strong Positive Correlation Y: Positive Correlation Z: Negative Correlation W: No Correlation

- Design Criterion A: The product needs to have an intuitive, clean user interface that supports several quality-of-life options and parameter definitions.

- Design Criterion B: The product needs to meet microphone industrial standards sampling frequency and operate well within design parameters posed by MATRIX module.
- Design Criterion C: The product needs to have reliable localization and tracking algorithms that meet the accuracy specifications outlined in table 1.
- Design Criterion D: The product needs to minimize costs and must not exceed 1500$.

# 4. Design Approach and Details

## 4.1 Design Concept Ideation, Constraints, Alternatives, and Tradeoffs

The core product of this project is an active localization algorithm software that provides accurate acoustic multi-source localization for various purposes including emergency rescue, speech localization, automatic surveillance camera, etc.

The design needs to accomplish:

1. Locate the acoustic source's relative position with respect to the microphone array when the source is a known sequence and provide direction information when the source is random speech signals.
2. Incorporate tracking algorithm when the source is moving.
3. Have an operation range of at least 20 meters.
4. Update and provide real-time visual demonstration of location on GUI.

Potential improvements if the above requirements are met:

1. Provide reliable multi-source localization
2. Provide reliable distance information when the incoming signal is a random acoustic signal of arbitrary strength

### Software

In modern day communication systems, active localization can be divided into range-based and range-free methods. Due to the nature of acoustic localization, we are only considering range-based algorithms since range-free algorithms require the deployment of anchor points. Among range-based algorithms, we have considered Time of Arrival (TOA), Time difference of Arrival (TDOA), Receive signal strength (RSS) and interferometric methods. Their tradeoffs and constraints are discussed below:

- TOA: TOA requires a transmitter and receiver to share the same clock, which is not practical for spontaneous speech source localization, but it is possible for wave generators, and it's easier to implement than other methods. We decided to keep this as the final back up plan if our main approach failed.
- TDOA: TDOA does not require timing information from the transmitter and uses multiple microphones in an array structure to extract phase information from the difference in time of the arrival signals. TDOA suffers from estimation error due to limited sampling rate of the microphone, causing triangulation to have a large error margin.
- RSS: RSS uses the signal strength of the receive signals to calculate the distances between the source and receiver, RSS provides more direct information on distance but suffers from the fact that multiple microphones have small spacing, thus the angle information cannot be deduced.
- TDOA-RSS: This is a method that combines both TDOA and RSS to estimate angles and distances separately to address the limitation of both methods.

We decided to use TDOA-RSS as the primary approach. The algorithm first calculates the time difference of arrival between multiple microphone pairs, performs generalized cross-correlation on signal sets, and uses a novel triangulation method to calculate the unit vector. Then the algorithm estimates the receive signal strength and, based on the pathloss model we developed in an indoor environment, gives information on the distance of the source.

### Hardware

The Hardware concerns the arrangement of microphone array and in what form the result will be presented. The team has considered the following methods:

- To implement the microphone array on a PCB that is mounted on a portable device along with a core microprocessor.
- To implement the microphone array on a skeleton structure with individual microphones' relative position easily changeable.
- To use the MATRIX VOICE module interfaced with Raspberry Pi3.

## 4.2 Preliminary Concept Selection and Justification

The selection process is both experimental and conceptual:

We decided our approach first by conceptually agreeing on the topics most members of the group are comfortable with. These preliminary decisions are:

- We decided in favor of acoustic signal localization as opposed to RF or Wi-Fi signal localization. The reasons are concrete: acoustic signal provides more utility in indoor environments whereas RF signal applies to larger environments; although Wi-Fi indoor localization is also very also widely applied, it is over researched with the abundance of survey papers available; acoustic signal is less researched and can be more tractable than Wi-Fi; lastly, the team in general has more exposure to acoustic signals than RF.
- We decided to prioritize tracking and single-source localization, while regarding multi-source as a feature we could implement if the first two options are met. This is because multi-source localization can be challenging, and we do not want to include it in the requirements.
- We decide to prioritize 2-D tracking over 3-D, because of the limitation of MATRIX VOICE module with no z-axis microphone component. In addition, during simulation in MATLAB, the elevation of the human will in fact only minimally affect the angle estimation. If required, we can also have elevation difference as an input to account for the error.

## Software

We then came down to deciding the exact overarching algorithm to go for based on a decision matrix. The performance is measured on a scale of 1-5 with 5 being the best. TOA, TDOA, RSS's relative difficulty for Wi-Fi has been established in *Liu-Survey on Wi-Fi,* and we adjusted the metric for acoustic signals.

Table 3. Decision Matrix of all Active Localization method considered

| | TOA | TDOA | RSS | Interferometric | TDOA-RSS |
|---|---|---|---|---|---|
| Easiness to implement | 4 | 3 | 2 | 2 | 3 |
| Accuracy | 4 | 3 | 2 | 5 | 4 |
| Novelty | 2 | 2 | 3 | 4 | 4 |
| Adaptability | 1 | 4 | 3 | 1 | 4 |

Therefore, we decided that TDOA-RSS would be the best approached followed by TDOA and TOA.

The backup plan if the TDOA-RSS approach does not work or does not meet the specifications is we will decide to use TOA. TOA method assumes the transmitter and receiver share the same clock, and we will be using two MATRIX arrays to triangulate the relative position of the acoustic source. This approach has reduced complexity and is almost guaranteed to work with more desirable accuracy, but it has severe limitations on the scope since clock information is required from the transmitter.

## Hardware

For the hardware part, the team quickly wrote off the idea of a portable device. This is because the algorithm is dependent on microphone spacing, and any small inaccuracies in the PCB design will have catastrophic consequences in terms of estimation error.

The skeleton structure with individual microphones is more practical. However, it causes more problems with building the structure, requiring individual microphones and being exceedingly difficult to handle. The team also spent considerable time doing research on the geometries of the array and evaluated their tradeoffs:

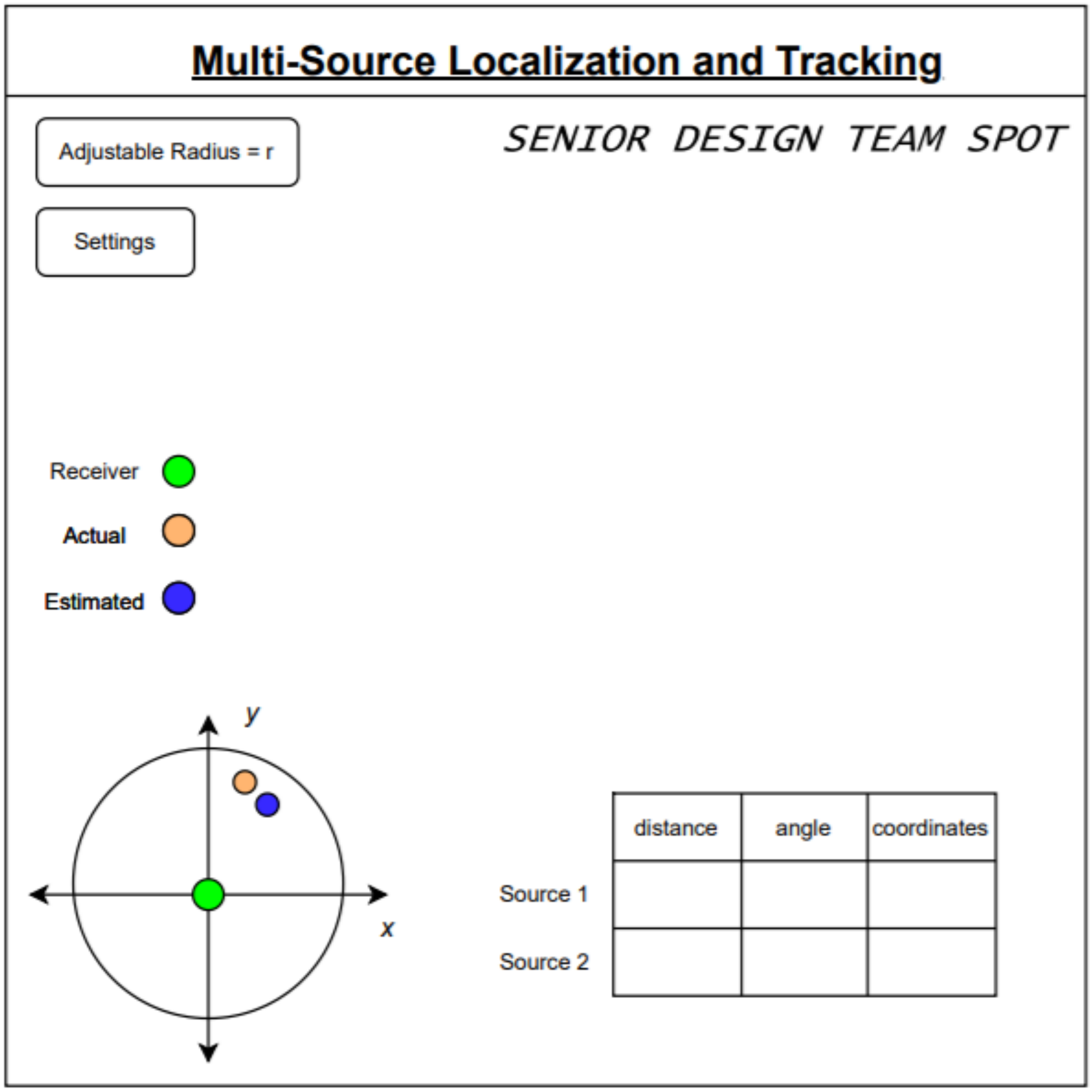
- 2-D linear array: Linear array is easy to implement but has the serious drawback of not being able to distinguish any source from its XY or YZ plane reflections. It also does not support any 3-D utilities.
- 3-D cross array: This array consists of five microphones on XY plane forming a cross with a z-axis microphone. This geometry is popular in literature but requires building individual microphones with careful spacing measurements.
- 2-D circular array: Circular array is better than linear array but also lacks 3-D utility, though it is possible.

During the simulation, we have discovered that for practical use, the elevation of the human body will not affect the direction estimation that much even with 2-D algorithms. Therefore, we decided to use a 2-D circular array in the form of MATRIX VOICE module.

The project will have a GUI that incorporates:

1. Primary display of the room that we are testing in with Estimated and Actual relative position
2. List of Distance/Direction
3. Parameter initialization (change the size of the room, microphone array position)
4. Start/Stop (UI), disable (distance/direction)
5. QoL features (color, brightness)
6. User Manuel (FAQ)

Below is a preliminary concept layout of the GUI, which is going to change.



| | distance | angle | coordinates |
|---|---|---|---|
| Source 1 | | | |
| Source 2 | | | |

Figure 2. Preliminary GUI diagram

## 4.3 Engineering Analyses and Experiment

The analysis is done with MATLAB simulation using the phased array toolbox in a free space propagation environment with FM wave.
The simulation has demonstrated the viability of 1. Cross-Correlation Method 2. Triangulation method 3. RSS calculation.

1. The cross-correlation method used is Generalized Cross-Correlation with Phase transform. This method achieves great accuracy in a free-space simulation environment and 96K sampling frequency, the widely regarded industry standard microphone sampling rate, and is supported by the MATRIX VOICE module. Attached is a simulation result of 8 pairs of distance difference of arrival of microphones. The microphones are labeled 1 to 6 and are positioned in terms of relative positioning axis on (0,0,0), (d,0,0), (0, d,0), (-d,0,0), (0, -d,0), (0,0, d), where d is the microphone spacing with a value of 20cm.

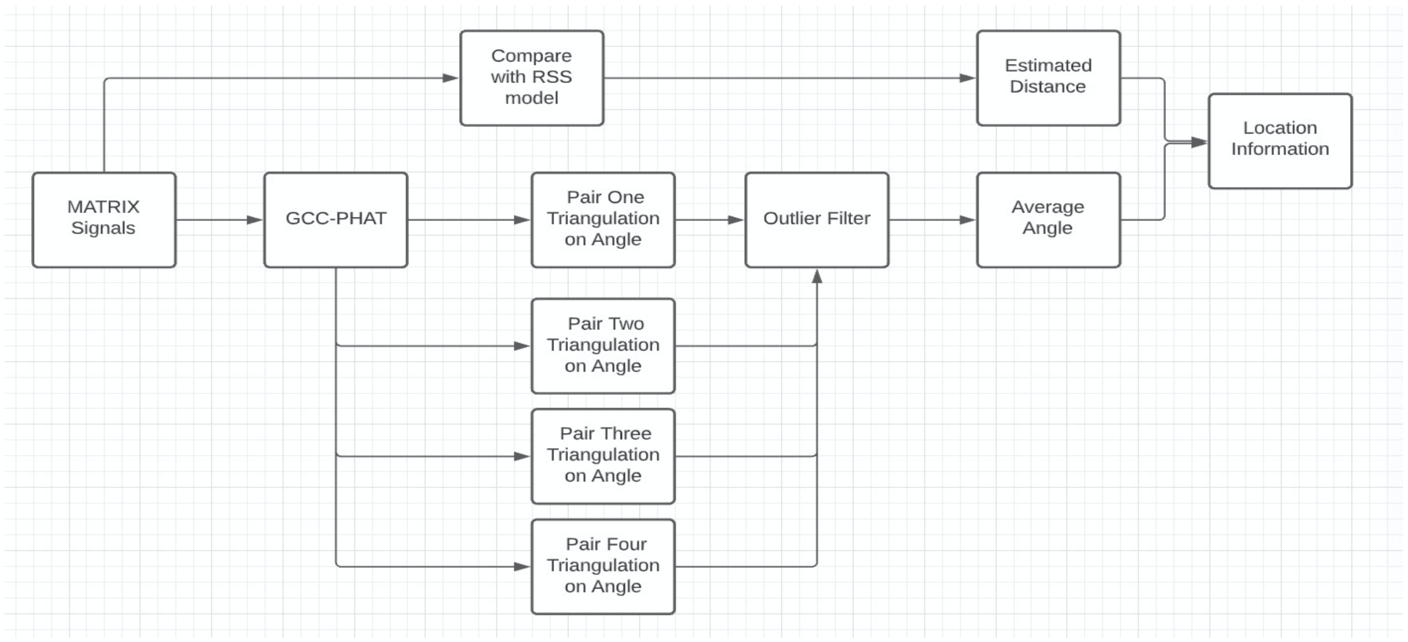
Table 3. Estimated and actual difference of distance of arrival

| | D12 | D13 | D14 | D15 | D16 | D23 | D24 | D25 |
|---|---|---|---|---|---|---|---|---|
| Estimated | 0.1842 | 0.2833 | -0.2054 | -0.2975 | 0.1842 | 0.0992 | -0.3896 | -0.4817 |
| Actual | 0.1864 | 0.2863 | -0.2013 | -0.2955 | 0.1864 | 0.0999 | -0.3877 | -0.4819 |

The GCC method has particularly good accuracy. Therefore, we decided to implement it. In terms of weighting factor, no weighting, phase transform, smoothed coherence transform, or ROTH are tested. As a result, phase transform provided the best performance in terms of peak heightening and noise reduction. We therefore decide to implement GCC-PHAT.

2. There are several triangulation methods that we explored. There are: direct form calculation, far-field + direct form, double far-field simplification, and averaged trigonometry methods. Triangulation is dependent on the real environment; we therefore need to test several different methods on the MATRIX. Based on the MATLAB simulation, the averaged trigonometry method attains an average direction error of 2 degrees, which is considered acceptable. We therefore decided to move along with the method.

3. The RSS method to measure distance information is not yet tested, but, conceptually, the free space pathloss model can be easily established. We decided to extract the signal strength by taking the squared mean of the first 100 samples, which was not technically challenging. Hence, we do not see it as a major concern.



Compare with RSS model
Estimated Distance
Location Information
MATRIX Signals
GCC-PHAT
Pair One Triangulation on Angle
Outlier Filter
Average Angle
Pair Two Triangulation on Angle
Pair Three Triangulation on Angle
Pair Four Triangulation on Angle

Figure 3. Preliminary Software Flowchart

### 4.4 Codes and Standards

The project will have two coding components. The software simulation will be done in MATLAB and will be translated into C++ for MATRIX HAL library. The GUI will be made using Python. The final project report and references will follow IEEE standard.

## 5. Project Demonstration

The project will be demonstrated by showing the estimation statistics obtained previously from an indoor setting, using the GUI and videos from a laptop or another computing device. We will show how the user will be able to adjust the distance that they want the device to function at, as well as the direction that they want to test in. Our device is relatively small, and only needs to be placed in a room to function. For the demonstration, we will be using estimation statistics from both speech signals and known frequency acoustic signals. The estimation results will contain signals at various frequencies, ranging from lower frequencies to 48 kHz to demonstrate our desired frequency range. We will also make sure the operation range is 20 meters, which is a far enough distance for our project to be of practical use. The demonstration will also include videos of successful tracking algorithm displays, to ensure that the microphone array can track the signal's movement. The GUI will show the actual distance and angle of our signal from the microphone array and will also display the experimental results as data is collected. This will allow us to compare the two and ensure that the experimental distance differs from the actual by no more than 30 centimeters and the recorded angle is accurate to within 3 degrees.

## 6. Schedule, Tasks, and Milestones:

| SENIOR DESIGN SCHEDULE | Week | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Task** | | | | | | | | | | | | | |
| PROPOSAL | | ALL | | | | | | | | | | | |
| PROJECT SUMMARY | | ALL | | | | | | | | | | | |
| GUI DESIGN | | | ALL (GUI) | | | | | | | | | | |
| ORDER 2nd MATRIX | | | HARRY | | | | | | | | | | |
| Design Review Presentation | | | | | ALL | ALL | 🟥 | | | | | | |
| Update Project Summary | | | | | ALL | ALL | 🟥 | | | | | | |
| Microphone Test | | | AJ | | | | | | | | | | |
| Far-Field Test | | | AJ + SIDONG | | | | | | | | | | |
| Decide on Plan A or B | | | AJ + SIDONG | | | | | | | | | | |
| Build GUI (2D PLANE) | | | Daniel | GUI(TBD) | GUI(TBD) | GUI(TBD) | | | | | | | |
| Build GUI (Display Data) | | | Tiffany | GUI(TBD) | GUI(TBD) | GUI(TBD) | | | | | | | |
| Build GUI (QoL Features) | | | Andrew | GUI(TBD) | GUI(TBD) | GUI(TBD) | | | | | | | |
| Test GUI | | | | | | | GUI(TBD) | GUI(TBD) | | | | | |
| SINGLE SOURCE TEST | | | | AJ+SIDONG | AJ+SIDONG | | | | | | | | |
| MULTI SOURCE TEST | | | | | | AJ+SIDONG | AJ+SIDONG | | | | | | |
| TRACKING | | | | | AJ+SIDONG | AJ+SIDONG | AJ+SIDONG | | | | | | |
| INTERFACE GUI WITH ALGO | | | | | | | | GUI(TBD) | GUI(TBD) | | | | |
| CAPSTONE DESIGN EXPO | | | | | | | | ALL | ALL | ALL | ALL | 🟥 | |
| FINAL DEMO | | | | | | | | ALL | ALL | ALL | ALL | | 🟥 |
| FINAL REPORT | | | | | | | | ALL | ALL | ALL | ALL | | 🟥 |
| Update Project Summary | | | | | | | | ALL | ALL | ALL | ALL | | 🟥 |

## 7. Cost Analysis

We have obtained a MATRIX VOICE (Xilinx Spartan 6 XC6SLX9 FPGA and 8MEMSMP34DB02 audio sensor digital microphones) from Dr. Anderson. SPOT is requesting a budget of $90 to purchase another microphone array, as one microphone array is not enough for the backup TOA localization algorithm. In terms of research, the team has put about 12 to 15 hours per week towards the project. The algorithm has been created and tested for approximately 4 to 6 hours per week from Sidong's observation. We have plans to create a GUI—graphical user interface—for the algorithm, so the team has begun to learn Python and utilize this language to develop a GUI, which is about 4 hours per week. The cost of labor and research regarding 20 to 25 hours per week is in the ballpark of—if we say $40 per hour—$800 to $1,000 per week.

## 8. Current Status

Our main objectives for the next week include testing the MATRIX Voice microphone array by translating our MATLAB based algorithm into C++. We are 40% completed with this task as the code is already written and we have scheduled time to conduct this test. We decided to first implement time difference estimation followed by angle calculation. Lastly, the RSS model will be established and tested. SPOT's other sub-team is focused on GUI design. We are in the preliminary stages of this task and are 20% complete. The remaining portion of this task will be complete once major subtasks are partitioned and assigned to team members.

## 9. Leadership Roles

1. **Tiffany Ho:**
    - Group leader – Collaborates with others to create a schedule with roles based on strengths and weaknesses of every individual on the GUI team.
    - Documentation Coordinator – Documents attendance for every meeting along with a few notes for project specifics from Dr. Ma.
2. **Daniel Scarborough:**
    - Webmaster – Documents the webmaster portion of our project.
3. **Ajeetpal Dhillon:**
    - Documentation Coordinator – Takes notes and documents specific details for every meeting.

- Software/Algorithms Lead – Collaborates with another software lead on creating the algorithm for this project.

4. **Harry Nguyen:**
   - Financial Manager – Responsible for keeping track of all finances along with order forms to be submitted to the ECE department for parts that are necessary for the project.
5. **Sidong Guo:**
   - Software/Algorithms Lead – Modifies the MATLAB algorithm and raspberry pi along with the matrix voice module and research methods of algorithms for our project. Collaborates with Ajeetpal Dhillon for this sub team.
     - Creates a separate schedule for himself and Ajeetpal Dhillon.
6. **Andrew Dulaney:**
   - Hardware Lead – Responsible for obtaining necessary hardware resources on campus before having to go to the ECE department to order parts.

## 10. References

1. M. D. Gillette and H. F. Silverman, "A Linear Closed-Form Algorithm for Source Localization From Time-Differences of Arrival," in IEEE Signal Processing Letters, vol. 15, pp. 1-4, 2008, doi: 10.1109/LSP.2007.910324.

2. L. Li, Y. Wang, X. Ma, C. Chen and X. Guan, "Dual-tone radio interferometric positioning systems for multi-target localization using a single mobile anchor," in China Communications, vol. 12, no. 1, pp. 25-35, Jan. 2015, doi: 10.1109/CC.2015.7084381.

3. M. Shinotsuka, Y. Wang, X. Ma and G. T. Zhou, "Designing radio interferometric positioning systems for indoor localizations in millimeter wave bands," 2014 48th Asilomar Conference on Signals, Systems and Computers, 2014, pp. 1184-1188, doi: 10.1109/ACSSC.2014.7094645.

4. C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 4, pp. 320-327, August 1976, doi: 10.1109/TASSP.1976.1162830.

5. J. Hu, C. Tsai, C. Chan and Y. Chang, "Geometrical arrangement of microphone array for accuracy enhancement in sound source localization," 2011 8th Asian Control Conference (ASCC), 2011, pp. 299-304.

6. G. Xiong, X. Li, X. Wu and Y. Ou, "A closed-form approach to acoustic source localization," 2012 IEEE International Conference on Information and Automation, 2012, pp. 739-744, doi: 10.1109/ICInfA.2012.6246916.

7. M. Shinotsuka, Lin Cheng, X. Ma, G. K. Chang and G. T. Zhou, "Hardware implementation of the space-time radio interferometric positioning system," 2015 IEEE International Wireless Symposium (IWS 2015), 2015, pp. 1-4, doi: 10.1109/IEEE-IWS.2015.7164550.

## Appendix

Matlab Simulation Codes to Justify Method Selections

```
clear;clc;close all;
%% Source Localization with Microphone array
% Method: Generalized Cross-correlation with The phase transform and
% far-field simplification

% Author: Sidong Guo
% Date: Feb 3rd 2022
%% Initialization

r=38.2978/1000; %In milimeter, radius of the circular array%
rx=8;ry=8;rz=0; %Source Location
rxULA = phased.OmnidirectionalMicrophoneElement;
rxpos1 = [0;0;0];
rxvel1 = [0;0;0];
rxax1 = azelaxes(0,0);
rxpos2 = [-38.13;3.58;0]/1000;
rxvel2 = [0;0;0];
rxax2 = azelaxes(0,0);
rxpos3 = [-20.98;32.04;0]/1000;
rxvel3 = [0;0;0];
rxax3 = azelaxes(0,0);
rxpos4 = [11.97;36.38;0]/1000;
rxvel4 = [0;0;0];
rxax4 = azelaxes(0,0);
rxpos5 = [35.91;13.32;0]/1000;
rxvel5 = [0;0;0];
rxax5 = azelaxes(0,0);
```

```
rxpos6 = [32.81;-19.77;0]/1000;
rxvel6 = [0;0;0];
rxax6 = azelaxes(0,0);
rxpos7 = [5;-37.97;0]/1000;
rxvel7 = [0;0;0];
rxax7 = azelaxes(0,0);
rxpos8 = [-26.57;-27.58;0]/1000;
rxvel8 = [0;0;0];
rxax8 = azelaxes(0,0);

srcpos = [rx;ry;rz];
srcvel = [0;0;0];
srcax = azelaxes(0,0);
srcULA = phased.OmnidirectionalMicrophoneElement;

% Parameters definition

Carrier_Fre = 96e3;             % 20 kHz
Propagation_Speed = 340;            % 340 m/s
Operating_range = 34;           % 34 m
pri = (2*Operating_range)/Propagation_Speed;
prf = 1/pri;
bw = 5e3;             % 5 kHz
fs = 9.6e4;           % 96 kHz
waveform = phased.LinearFMWaveform('SampleRate',fs,'SweepBandwidth',bw,'PRF',prf,'PulseWidth',pri/10);
signal = waveform();

radiator = phased.WidebandRadiator('Sensor',srcULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector1 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector2 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector3 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector4 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector5 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector6 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector7 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);
collector8 = phased.WidebandCollector('Sensor',rxULA,'PropagationSpeed',Propagation_Speed,'SampleRate',fs,...
    'CarrierFrequency',Carrier_Fre);

channel1 = phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel2 = phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel3= phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel4 = phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel5 = phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel6 = phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel7= phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
channel8 = phased.WidebandFreeSpace('PropagationSpeed',Propagation_Speed,...
    'SampleRate',fs,'OperatingFrequency',Carrier_Fre);
```

```
[~,ang1_transmit] = rangeangle(rxpos1,srcpos,srcax);
[~,ang2_transmit] = rangeangle(rxpos2,srcpos,srcax);
[~,ang3_transmit] = rangeangle(rxpos3,srcpos,srcax);
[~,ang4_transmit] = rangeangle(rxpos4,srcpos,srcax);
[~,ang5_transmit] = rangeangle(rxpos5,srcpos,srcax);
[~,ang6_transmit] = rangeangle(rxpos6,srcpos,srcax);
[~,ang7_transmit] = rangeangle(rxpos7,srcpos,srcax);
[~,ang8_transmit] = rangeangle(rxpos8,srcpos,srcax);

sig_transmit = radiator(signal,[ang1_transmit ang2_transmit ang3_transmit ang4_transmit ang5_transmit
ang6_transmit ang7_transmit ang8_transmit]);

sigp1 = channel1(sig_transmit(:,1),srcpos,rxpos1,srcvel,rxvel1);
sigp2 = channel2(sig_transmit(:,2),srcpos,rxpos2,srcvel,rxvel2);
sigp3 = channel3(sig_transmit(:,3),srcpos,rxpos3,srcvel,rxvel3);
sigp4 = channel4(sig_transmit(:,4),srcpos,rxpos4,srcvel,rxvel4);
sigp5 = channel5(sig_transmit(:,5),srcpos,rxpos5,srcvel,rxvel5);
sigp6 = channel6(sig_transmit(:,6),srcpos,rxpos6,srcvel,rxvel6);
sigp7 = channel7(sig_transmit(:,7),srcpos,rxpos7,srcvel,rxvel7);
sigp8 = channel8(sig_transmit(:,8),srcpos,rxpos8,srcvel,rxvel8);

[~,ang1_receive] = rangeangle(srcpos,rxpos1,rxax1);
[~,ang2_receive] = rangeangle(srcpos,rxpos2,rxax2);
[~,ang3_receive] = rangeangle(srcpos,rxpos3,rxax3);
[~,ang4_receive] = rangeangle(srcpos,rxpos4,rxax4);
[~,ang5_receive] = rangeangle(srcpos,rxpos5,rxax5);
[~,ang6_receive] = rangeangle(srcpos,rxpos6,rxax6);
[~,ang7_receive] = rangeangle(srcpos,rxpos7,rxax7);
[~,ang8_receive] = rangeangle(srcpos,rxpos8,rxax8);

sigr1 = collector1(sigp1,ang1_receive);
sigr2 = collector2(sigp2,ang2_receive);
sigr3 = collector3(sigp3,ang3_receive);
sigr4 = collector4(sigp4,ang4_receive);
sigr5 = collector5(sigp5,ang5_receive);
sigr6 = collector6(sigp6,ang6_receive);
sigr7 = collector7(sigp7,ang7_receive);
sigr8 = collector8(sigp8,ang8_receive);

xcor12_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr2)))./(abs(fft(sigr1)).*abs(fft(sigr2))))));
xcor13_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr3)))./(abs(fft(sigr1)).*abs(fft(sigr3))))));
xcor14_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr4)))./(abs(fft(sigr1)).*abs(fft(sigr4))))));
xcor15_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr5)))./(abs(fft(sigr1)).*abs(fft(sigr5))))));
xcor16_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr6)))./(abs(fft(sigr1)).*abs(fft(sigr6))))));
xcor17_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr7)))./(abs(fft(sigr1)).*abs(fft(sigr7))))));
xcor18_PHAT=abs(fftshift(ifft((fft(sigr1).*conj(fft(sigr8)))./(abs(fft(sigr1)).*abs(fft(sigr8))))));

Actual_Arrival=zeros(1,4);
Actual_Arrival(1)=sqrt(rx^2+ry^2+rz^2)/Propagation_Speed;
Actual_Arrival(2)=sqrt((rx+0.03813)^2+(ry-0.00358)^2+rz^2)/Propagation_Speed;
Actual_Arrival(3)=sqrt((rx+0.02098)^2+(ry-0.03204)^2+rz^2)/Propagation_Speed;
Actual_Arrival(4)=sqrt((rx-0.01197)^2+(ry-0.03638)^2+rz^2)/Propagation_Speed;

Actual_difference=zeros(1,3);
Actual_difference(1)=Actual_Arrival(1)-Actual_Arrival(2);
Actual_difference(2)=Actual_Arrival(1)-Actual_Arrival(3);
Actual_difference(3)=Actual_Arrival(1)-Actual_Arrival(4);

Peak_Value=zeros(1,7);
Estimated_difference=zeros(1,7);
```

```
for i=1:length(xcor12_PHAT)
    if xcor12_PHAT(i)>Peak_Value(1)
        Estimated_difference(1)=(i-0.1*fs-1);
        Peak_Value(1)=xcor12_PHAT(i);
    end
    if xcor13_PHAT(i)>Peak_Value(2)
        Estimated_difference(2)=(i-0.1*fs-1);
        Peak_Value(2)=xcor13_PHAT(i);
    end
    if xcor14_PHAT(i)>Peak_Value(3)
        Estimated_difference(3)=(i-0.1*fs-1);
        Peak_Value(3)=xcor14_PHAT(i);
    end
    if xcor15_PHAT(i)>Peak_Value(4)
        Estimated_difference(4)=(i-0.1*fs-1);
        Peak_Value(4)=xcor15_PHAT(i);
    end
    if xcor16_PHAT(i)>Peak_Value(5)
        Estimated_difference(5)=(i-0.1*fs-1);
        Peak_Value(5)=xcor16_PHAT(i);
    end
    if xcor17_PHAT(i)>Peak_Value(6)
        Estimated_difference(6)=(i-0.1*fs-1);
        Peak_Value(6)=xcor17_PHAT(i);
    end
    if xcor18_PHAT(i)>Peak_Value(7)
        Estimated_difference(7)=(i-0.1*fs-1);
        Peak_Value(7)=xcor18_PHAT(i);
    end
end
Estimated_difference=Estimated_difference/fs;
Distance_difference=Estimated_difference*Propagation_Speed;
Actual_Distance=Actual_difference*Propagation_Speed;

d12=Distance_difference(1);
d13=Distance_difference(2);
d14=Distance_difference(3);
d15=Distance_difference(4);
d16=Distance_difference(5);
d17=Distance_difference(6);
d18=Distance_difference(7);

%Distance_Matrix=[-38.13,3.58;-20.98,32.04]/1000;

%v=(1817.45*d13+1000*d12)/54.65;
%u=(3.58*v-1000*d12)/38.13;
theta2=acosd(abs(d12)/0.0383)-5.3637
theta3=90-(acosd(abs(d13)/0.0383)-33.217)
theta6=acosd(abs(d16)/0.0383)-20.3512
theta7=180-82.5-acosd(abs(d17)/0.0383)

average_theta=(theta3+theta2+theta6+theta7)/4;
if abs(theta2-average_theta)>10
    Estimated_angle=(theta3+theta6+theta7)/3;
elseif abs(theta3-average_theta)>10
    Estimated_angle=(theta2+theta6+theta7)/3;
elseif abs(theta6-average_theta)>10
    Estimated_angle=(theta2+theta3+theta7)/3;
elseif abs(theta7-average_theta)>10
    Estimated_angle=(theta2+theta6+theta3)/3;
else
```

```
    Estimated_angle=(theta3+theta2+theta6+theta7)/4;
end
Estimated_angle
real_angle=atand(ry/rx)
```