EEE3097S 2023 MILESTONE 2: FIRST PROGRESS REPORT

ACOUSTIC TRIANGULATION USING MULTILATERATION AND GENERAL CROSS-CORRELATION WITH PHASE TRANSFORM

15 September 2023

Simon Carthew - CRTSIM008
Robert Dugmore - DGMROB001
Si Teng Wu - WXXSIT001

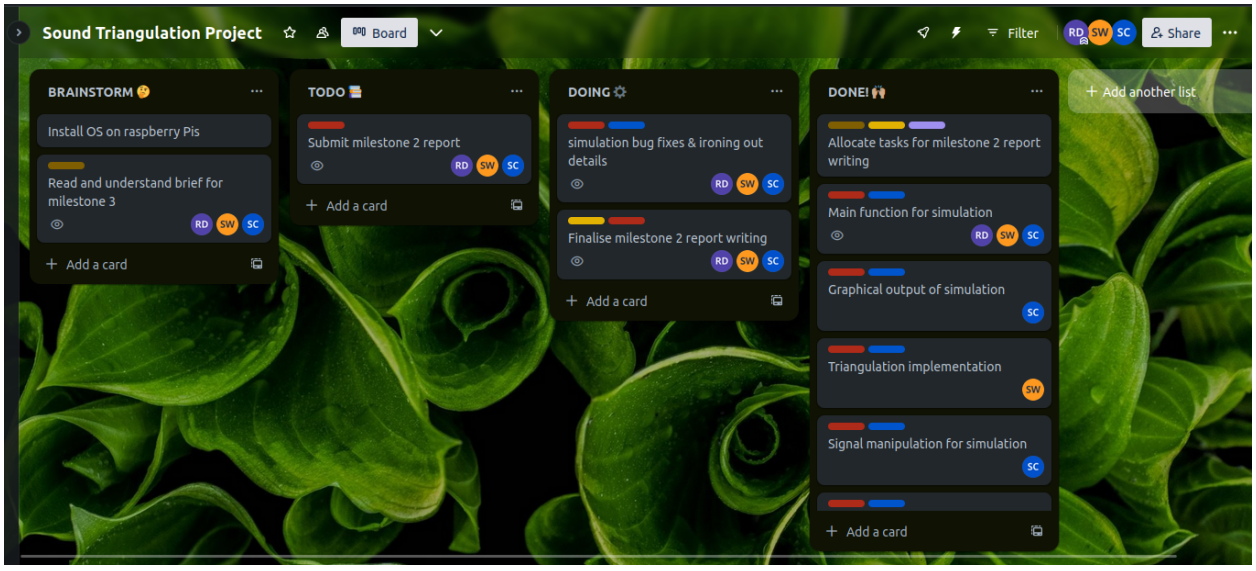
ADMIN DOCUMENTS

TABLE OF INDIVIDUAL CONTRIBUTIONS

The tasks listed below were only the initial tasks allocated to each person. Ultimately, the group met multiple times and gave extensive input to each other on all tasks listed.

Name	Sections
Simon Carthew	Coding: Signal processing, GUI, signal generation Report writing: Descriptions of coding sections, results, simulation, system setup
Robert Dugmore	Coding: GCC-PHAT, main simulation function Report writing: Analysis, challenges & limitations, next steps, conclusions
Si Teng Wu	Coding: Triangulation, GCC-PHAT Report writing: Descriptions of coding sections, GCC-PHAT algorithm, results, ATPs

SCREENSHOT OF TRELLO PAGE



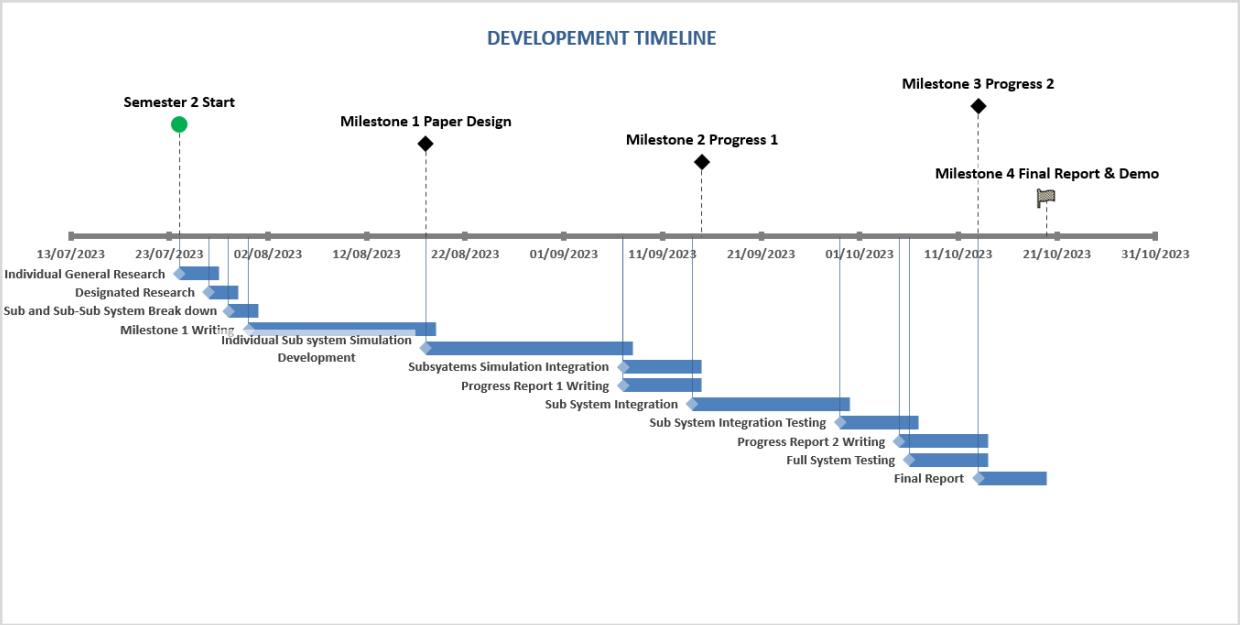
LINK TO GITHUB

https://github.com/rothdu/EEE3097S

The simulation is run by executing “simulate.py” from the “Simulation” directory. The results are recorded in the “Test Results” directory. Each run of the simulation appends new subdirectories in “Test Results”, i.e., old test results are not discarded unless they are manually deleted.

TIMELINE

Below is the updated project timeline:



The team has managed to keep up with all previously set goals. Some changes to the previous timeline are that the Milestone 2 deadline was shifted to the 15/09/2023 and the tasks completed before that were adjusted to be simulation based tasks. The Milestone 2 extension allowed for finer adjustments to be made to the simulation and progress report.

SIMULATION SETUP

SIMULATION ENVIRONMENT AND TOOLS

The simulation environment is a series of python scripts that work together to create a one-click run style environment where each time the simulation is run, a combination of set parameters and random conditions are used to produce outputs that are displayed in the form of plots and resultant excel spreadsheets. The set parameters tested are the four types of noise added to the original sound source as well as two different microphone placements which are set using a config file. The random conditions are the actual noise signals added to each microphone as well as the source point coordinates. Different python libraries were created to handle the various parts of the simulation. These parts include signal generation, graphical user interfacing, triangulation, GCC-phat and system execution. These parts all represent the whole or part of some of the subsystems from the paper design.

The tools used are various open-source python libraries which includes matplotlib, numpy, pandas, colorednoise, scipy, openpyxl and sympy as well as custom designed scripts including gui.py, signal_procesing.py, simulate.py, triangulate.py, gcc_phat.py and wav_signal.py. The final tool is a python run-time environment used to execute the main function of simulate.py which is, but not limited to, Visual Studio Code.

SIMULATION EXECUTION

The simulation is executed by running the main function of the simulate.py script. Inside the main function, 10 random points are generated and the test function from simulate.py is called four times each with different noise conditions. The test function is explained below.

Test Function:

The test function takes in two different microphone configurations, 10 source points and the type of noise being tested. The following is how the test function is executed:

- Source signal is loaded from a WAV file.
- Executed for each microphone configuration:
    - Executed for each point:
        - Sound signals generated for each microphone using the gen_delay function from wav_signal.py
        - The gcc_phat function from gcc_phat.py is used to calculate the estimated TDOA for all three microphone pairs.
        - The estimated TDOAs are used by the triangulation function from the triangulation.py script to calculate the estimated point location and generate the relevant hyperbolas
    - The results from all 10 points are inputted into the run function of the gui.py script which generates resultant spreadsheets and plots.

    - The results are saved to dynamically generated directories.

RATIONALE

The reasons from the chosen simulation environment and tools are as follows:

- The simulation files translate well to actual implementation of the design as many functions can be reused.
- Team members are familiar with Python which promotes faster development time.
- Parallel development is simple as simulation parts split up into independent python scripts.
- Better control over ATP results as all ATPs can be checked simultaneously with one execution.
- Test multiple input parameters with ease using simple iterable loops.

SIMPLIFICATIONS & ASSUMPTIONS

Simplifications:

- Mic positions are perfect.
- The resolution of the microphones is ignored, generated signals have arbitrary resolution.
- Only one type of noise present at a time.
- The source is not moving and moving effects are not considered.
- Pi Synchronisation issues are not evaluated.
- There is no difference in volume of the signal at each mic.

Assumptions:

- The noise at each microphone is the same type of noise but a different random set of noise.
- The time the signal takes to arrive from the Raspberry Pi's at the local computer is zero seconds.
- The captured noise at each microphone is captured at exactly the same real time.

SYSTEM DESIGN AND IMPLEMENTATION

WHOLE SYSTEM

Simulation:

The entire system is simulated using a python script called “simulation.py”. This script's main function is responsible for using functions from other libraries to simulate the system flow between all the subsystems. The different libraries used by the simulation script are gcc_phat.py, wav_signal.py, signal_processing.py, gui.py and triangulation.py which can all be considered as subsystems or at least part of a subsystem.

Experiments:

Each execution of the simulation script will test a set of 10 random points under varying conditions. The different condition parameters altered are noise types and mic positions. Combinations of these parameters are tested using iterating loops where in each iteration the system is used to locate the 10 points under its unique conditions and generates a new set of results.

SIGNAL ACQUISITION

Simulation:

For the purpose of the simulation, the sound that is to be located is a person snapping their fingers within the grid The sound source used can be found at Finger Snap Sound Effect - YouTube [1]. The sound is saved as a WAV file and later converted to a numpy array. The acquisition of this signal by each microphone is simulated using the gen_delay function from the wav_signal.py script that takes in the original signal, the source point coordinates, the location of the microphone that is capturing the sound and the microphone sampling rate.
The function returns a signal that is an extracted window of the original sound signal. The signal is shifted by the necessary sample offset. The sample offset is directly related to the time taken for the signal to arrive at the microphone that the signal is being generated for. Four different signals are generated for each microphone as the delay at each microphone will be different based on where the sound source is located.

To simulate the noise that may occur at each microphone, the add_noise function from the signal_processing.py library is used to add varying types of random noise to each microphone signal. The function takes in the original microphone signal and the type of noise to be added as parameters and returns the original signal with added noise. Independent sets of noise are added to each microphone to simulate the fact that the noise captured by each microphone may be slightly different based on its position. The system is tested using different types of noise

including impulse, gaussian and pink noise. The pink noise simulates background noise from people speaking, impulse noise simulates sudden loud conversation or equipment noise and gaussian noise simulates random variations in the ambient noise level.

Experiments:

Testing the system with ten random test points ensures that the system works when different signals are received at each microphone. The signals received at each microphone will be different for each point as the time delay at each microphone varies based on the sound source location.

The four conditions of pink, gaussian, impulse and no noise are tested with both microphone configurations to create 8 different sets of unique conditions. The results produced by the GUI simulation will give a metric of how the system behaves under varying noise conditions.

TDOA

Simulation:

A library for GCC-PHAT was obtained from here called gcc_phat.py that implements GCC-PHAT algorithm described in the TDOA Algorithm section [2].

Pairs of signals generated by the signal acquisition simulation are passed to the GCC-PHAT algorithm, which outputs an estimated time-difference of arrival (TDOA).

Algorithm:

The TDOA algorithm used is GCC-PHAT: Generalised Cross Correlation with Phase Transform. GCC-PHAT is a time delay estimation algorithm that extends the base cross correlation method by multiplying the Cross Power Spectral Density function by a weighting function, PHAT, before calculating the time delay. By doing so, the GCC-PHAT function will be left with only phase information, producing a delta function at the estimated delay while "pushing" other components outwards[3][4].

The algorithm of GCC-PHAT is as follows:

1. Given two signals $x_1(t)$ and $x_2(t)$ where $x_1(t)$ is the reference mic signal.
2. The Fourier Transform of each signal is taken, giving $X_1(\omega)$ and $X_2(\omega)$.
3. The conjugate of $X_2(\omega)$ is taken, giving $[X_2^*(\omega)]$
4. The Cross Power Spectral Density is calculated using:

$$G_{x1x2}(\omega) = X_2(\omega)[X_2^*(\omega)]$$

5. The weighting function is calculated and multiplied with the CPSD:

$$\Psi_{PHAT}(\omega) = \frac{1}{\left|X_2(\omega)[X_2^*(\omega)]\right|}$$

$$G_{x1x2-PHAT}(\omega) = \frac{X_2(\omega)[X_2^*(\omega)]}{\left|X_2(\omega)[X_2^*(\omega)]\right|}$$

6. Inverse Fourier Transform is taken giving the GCC-PHAT function:

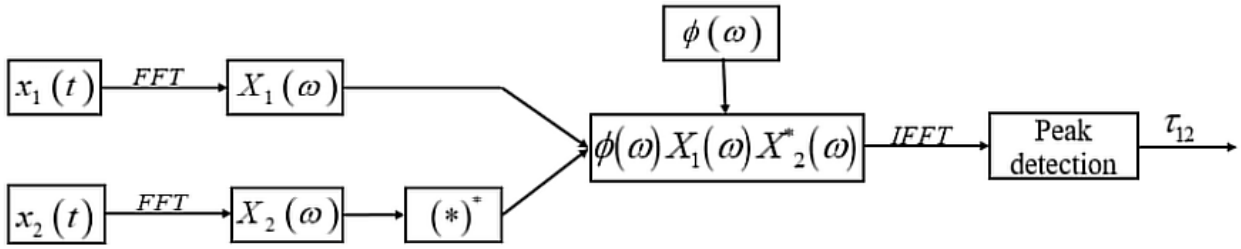
$$R_{x1x2-PHAT}(d) = IFFT\left\{\frac{X_2(\omega)[X_2^*(\omega)]}{\left|X_2(\omega)[X_2^*(\omega)]\right|}\right\}$$

7. The delay is the position of the highest peak in the GCC-PHAT functions:

$$d = argmax\left\{R_{x1x2-PHAT}(d)\right\}$$

Equations adapted from [3][4].

The following flow diagram demonstrates the algorithm.



Taken from [5]

Experiments:

An estimated TDOA is calculated for 10 points for each mic array and each noise condition. In each test, the microphone at (0, 0) is chosen as the reference mic, and signals from other microphones are compared to this one.

These are then compared against the known TDOA that was calculated in the signal acquisition phase. The difference between the actual TDOA and estimated TDOA is found and divided by the maximum possible TDOA to give a percentage change in the estimated TDOA. This figure is used by the relevant ATP to judge the performance of the gcc_phat function.

TRIANGULATION

Simulation:

The triangulation simulation function exists in the triangulation.py library. A triangulation function returns an estimated x and y position of the source, an x and y meshgrid representing the grid of the experiment and 3 hyperbolic curves used for graphing. The function accepts 4 sets of positions of the four mics (the first position being the reference-mic's position and the other 3 being the helper-mics), a set of 3 distances = (TDOA*speed of propagation) and parameters for the meshgrid.

Using the TDOA, a triangulation formula is used to find possible positions of the source based on a single mic pair:

$$\sqrt{(x - x_1)^2 - (y - y_1)^2} - \sqrt{(x - x_2)^2 - (y - y_2)^2} = c(\Delta t)$$

where x, y is the estimated source position, x_1, y_1, x_2, y_2 is the mic positions, c is the speed of signal propagation and Δt is the time delay [6].

Three hyperbolic curves were calculated for each ref-mic/ helper-mic pair and using Sympy, three intersections were found between the three hyperbolic curves. A check was added for invalid intersections such as complex results that arise from extreme TDOAs that give no real solutions or intersections outside of the specified grid space. A midpoint formula is used with the valid intersections to estimate the final estimated source position.

Experiments:

In simulation, two mic-arrays were tested to determine the best position for the mics: Four corners array - 4 mics on the corners of the grid or Inline array - 4 mics in a line along the x axis. Each mic-array was tested with all noise conditions and multiple tests were run under each condition.

The actual and estimated x and y source positions were compared in each case and evaluated. They are compared by finding the difference between the actual value and the estimated value and dividing by the maximum axis value to give a percentage change in the estimated value. This is then later used by the ATPs to validate whether the triangulation function is working correctly

GUI

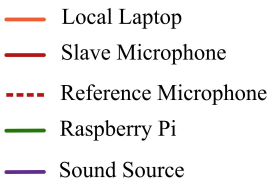
Simulation:

The GUI environment is simulated using the run function in the gui.py library. This function takes in the results from the system that are produced by each unique set of conditions. The function dynamically creates directories in the current working directory and saves the results in the next available directory. The results are in the form of plots showing the generated hyperbolas, actual source locations and estimated source locations for each test point. As well as excel spreadsheets showing the actual and estimated TDOA and source positions. In these excel spreads there are columns stating whether the relevant ATPs have been passed for that specific test point.

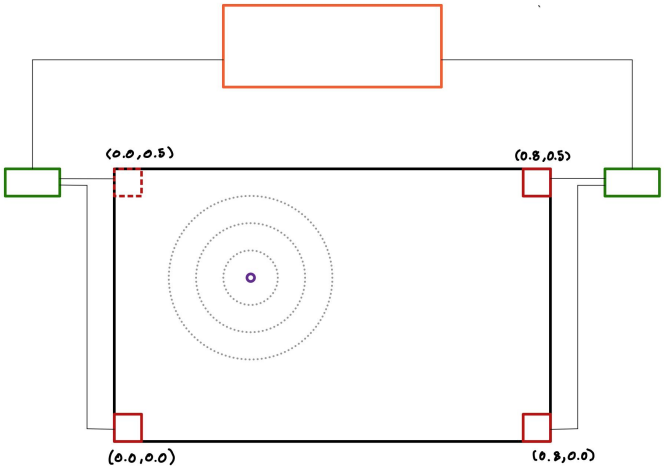
SYSTEM ARCHITECTURE:

Below are the two system architectures tested by the simulation. For each set of noise parameters, both system architectures are tested.

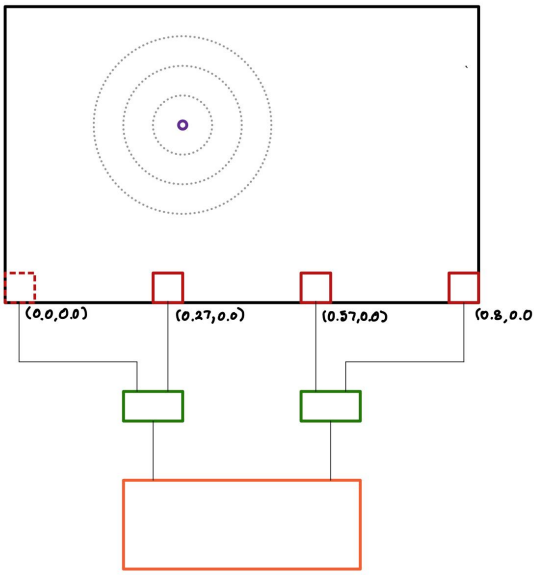
Figure Legend:



Four corners configuration:



Inline Microphone Configuration



SIMULATION RESULTS AND ANALYSIS

ATPs

	Figures of merit	Test Procedure	Acceptable performance
1	Same number of sound bites for all four microphones	Generate an array of timestamp data from the two Raspberry Pis over a number of interrupt triggers and see if the master and slave Pi output the same data in the arrays. This will show if the data collection will yield the same results as they are starting at the same time.	Four sound bites, no less, no more.
2	Every clock cycle, there is a full set of data	Use a 280 Hz sine wave sound source and check if each mic returns similar data every clock cycle.	Data same number of discrete points
3	Clock signal is 10 Hz	Connect the output of the clock signal to an oscilloscope and measure the frequency.	Clock signal is between 9 and 11 Hz
4	Microphones are able	Record a 5 Hz signal to each mic and	Data must have the

	to read a 10 second audio clip at 44.1 kHz sample frequency, 16 bit resolution	export the data to Python. Analyse the data for the correct number of discrete points, bit resolution and record time.	same number discrete points, bit resolution and between 10.1 and 9.9 seconds
5	A numpy discrete sound magnitude array is correctly collected by each mic	Play a 280 Hz signal to each mic and capture a few periods. Send data and plot in Python on the main computer to check for a valid sine wave. Test to be conducted at the maximum and minimum distance to the mics in order to test for saturation or insufficiency.	The output is a valid sine wave and the difference in amplitude at max and min distance is within 10%
6	Estimated TDoA matches theoretical TDoA	Create 4 identical numpy arrays with a 280 Hz sine wave and a delay on each. Input the array into the TDOA method and check results. Then use actual recorded data from mics and check results. Test is good if TDOA is within 1% error from theoretical value.	The TDOA not under noise conditions is within 1% tolerance
7	Estimated TDoA is still accurate under noisy conditions	Obtain recording of the lab when noisy and obtain a SNR ratio by doing a power spectrum analysis. Simulate the TDOA algorithm with a 280 Hz sine wave in 4 numpy arrays with different added noise and a known time shift	The TDOA under noise conditions is within 3% tolerance
8	Estimated x and y position from hyperbolas when source is moving at maximum speed of 0.1m/s.	Simulate a scenario of a moving signal and check if the algorithm is correct. Then create various TDOA samples with known points on the grid and compare results to actual position. Finally, move the sound source on known paths in the grid space and compare the estimated paths to the actual paths.	Accuracy is within 2% tolerance for x and y
9	Estimated x and y position from hyperbolas when source is moving at maximum speed of 0.1m/s.	Simulate a scenario of a moving signal and check if the algorithm is correct. Then create various TDOA samples with known points on the grid and compare results to actual position. Finally, move the sound source on known paths in the grid space and compare the estimated paths to the actual paths.	Accuracy is within 5% tolerance for x and y

10	Graphics updating correctly every 0.25 seconds	Feed dummy data to the UI and test if it correctly displays everything to check performance and accuracy. The test is passed if the data is correct, and appears within expected time.	Updates occur at 1 second max
11	User interface receives qualitative approval on the basis of useability from different people	Ask others to try to use the UI and obtain feedback and qualitative approval	At least 5 different approval ratings
12	All buttons on user interface perform expected behaviour when clicked	Create a checklist of each functionality and test each. Check fluidity between each button/functionality to ensure no crash/freeze conditions. Test is passed if the checklist is complete.	Every button is checked and passes, no ghost buttons

Tested summary:

TDOA:

6. Estimated TDoA matches theoretical TDoA within 1% tolerance.
7. Estimated TDoA is still accurate under noisy conditions within 3% tolerance.

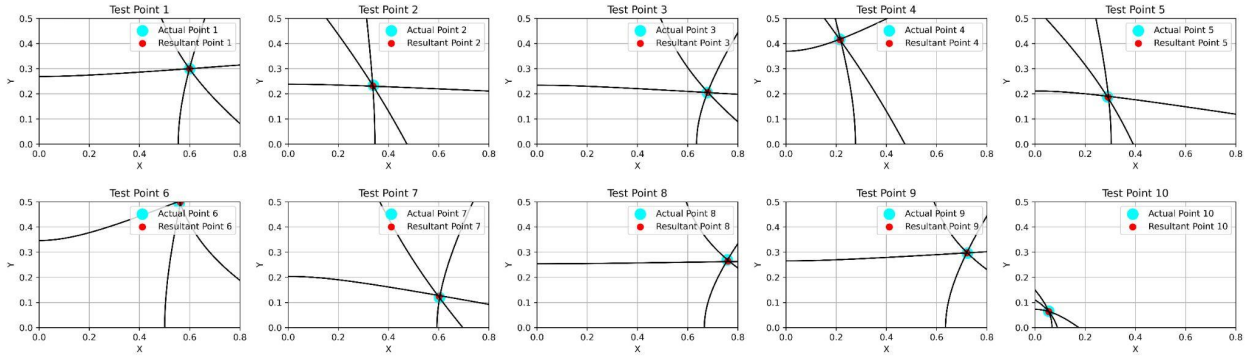
Triangulation:

8. Estimated x, y position is within 2% tolerance when source is moving at a maximum speed of 0.1m/s.
9. Estimated x, y position is within 5% tolerance under noisy conditions when the source is moving at maximum speed of 0.1m/s

RESULTS FOR MICROPHONES AT 4 CORNERS

No Noise:

Plotted Results:



Triangulation Results

Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.59890,0.30008)	0.13796	-0.01630	Pass
2	(0.33974,0.23381)	(0.33802,0.23158)	0.21510	0.44584	Pass
3	(0.67751,0.20546)	(0.68000,0.20579)	-0.31090	-0.06738	Pass
4	(0.21401,0.41756)	(0.21612,0.41527)	-0.26378	0.45906	Pass
5	(0.28929,0.1877)	(0.29095,0.18694)	-0.20741	0.15297	Pass
6	(0.55774,0.49462)	(0.56190,0.50015)	-0.51930	-1.10581	Pass
7	(0.60302,0.11883)	(0.60151,0.12436)	0.18868	-1.10629	Pass
8	(0.75757,0.26965)	(0.76036,0.26477)	-0.34943	0.97607	Pass
9	(0.72261,0.29596)	(0.72082,0.29770)	0.22318	-0.34799	Pass
10	(0.05471,0.06519)	(0.05454,0.06388)	0.02185	0.26207	Pass

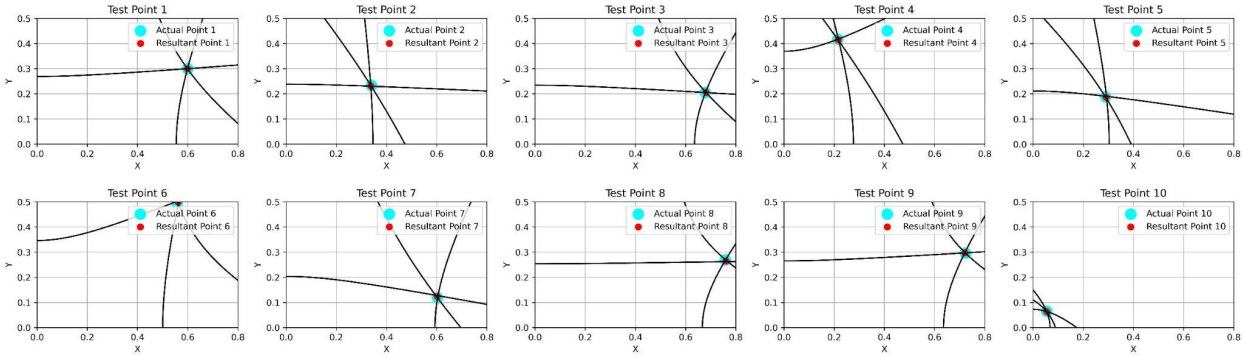
TDOA Results:

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.15183	0.20539	-0.02569	Pass
2	0.45632	0.45549	0.42511	Pass
3	-0.23514	-0.51157	0.01023	Pass

4	-0.34767	0.10574	0.05518	Pass
5	-0.66499	0.10964	-0.38822	Pass
6	-0.64519	-0.68777	-0.56855	Pass
7	-0.15842	0.02738	-0.65325	Pass
8	-0.17031	-0.15589	0.49026	Pass
9	0.42376	-0.10569	-0.09358	Pass
10	0.10468	0.22167	0.21673	Pass

Pink Noise

Plotted Results:



Triangulation Results:

Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.59890,0.30008)	0.13796	-0.01630	Pass
2	(0.33974,0.23381)	(0.33802,0.23158)	0.21510	0.44584	Pass
3	(0.67751,0.20546)	(0.68000,0.20579)	-0.31090	-0.06738	Pass
4	(0.21401,0.41756)	(0.21612,0.41527)	-0.26378	0.45906	Pass
5	(0.28929,0.1877)	(0.29095,0.18694)	-0.20741	0.15297	Pass
6	(0.55774,0.49462)	(0.56190,0.50015)	-0.51930	-1.10581	Pass
7	(0.60302,0.11883)	(0.60151,0.12436)	0.18868	-1.10629	Pass

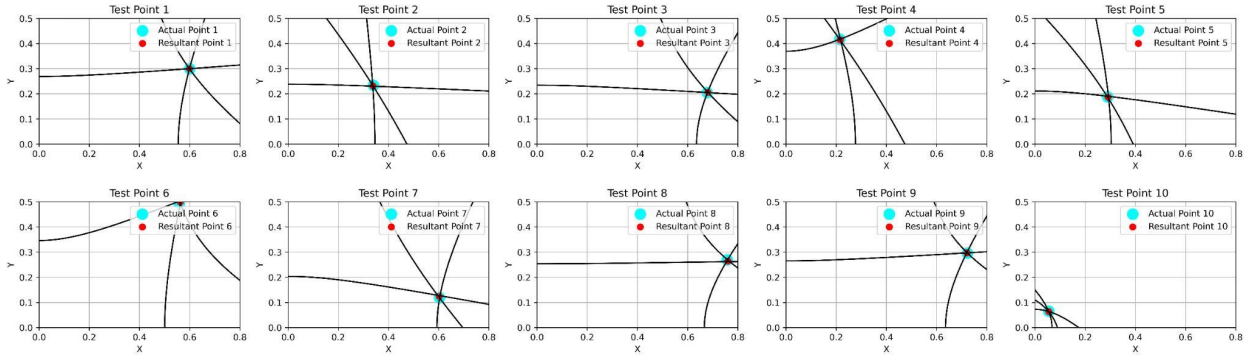
8	(0.75757,0.26965)	(0.76036,0.26477)	-0.34943	0.97607	Pass
9	(0.72261,0.29596)	(0.72082,0.29770)	0.22318	-0.34799	Pass
10	(0.05471,0.06519)	(0.05454,0.06388)	0.02185	0.26207	Pass

TDOA Results:

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.15183	0.20539	-0.02569	Pass
2	0.45632	0.45549	0.42511	Pass
3	-0.23514	-0.51157	0.01023	Pass
4	-0.34767	0.10574	0.05518	Pass
5	-0.66499	0.10964	-0.38822	Pass
6	-0.64519	-0.68777	-0.56855	Pass
7	-0.15842	0.02738	-0.65325	Pass
8	-0.17031	-0.15589	0.49026	Pass
9	0.42376	-0.10569	-0.09358	Pass
10	0.10468	0.22167	0.21673	Pass

Gaussian Noise

Plotted Results:



Triangulation Results:

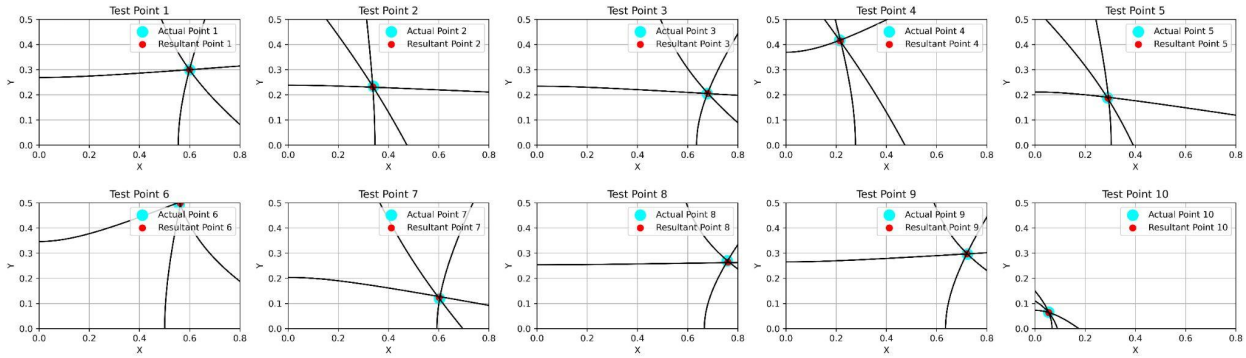
Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.59890,0.30008)	0.13796	-0.01630	Pass
2	(0.33974,0.23381)	(0.33802,0.23158)	0.21510	0.44584	Pass
3	(0.67751,0.20546)	(0.68000,0.20579)	-0.31090	-0.06738	Pass
4	(0.21401,0.41756)	(0.21612,0.41527)	-0.26378	0.45906	Pass
5	(0.28929,0.1877)	(0.29095,0.18694)	-0.20741	0.15297	Pass
6	(0.55774,0.49462)	(0.56190,0.50015)	-0.51930	-1.10581	Pass
7	(0.60302,0.11883)	(0.60151,0.12436)	0.18868	-1.10629	Pass
8	(0.75757,0.26965)	(0.76036,0.26477)	-0.34943	0.97607	Pass
9	(0.72261,0.29596)	(0.72082,0.29770)	0.22318	-0.34799	Pass
10	(0.05471,0.06519)	(0.05454,0.06388)	0.02185	0.26207	Pass

TDOA Results:

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3% Error	Pass/Fail
1	0.15183	0.20539	-0.02569	Pass
2	0.45632	0.45549	0.42511	Pass
3	-0.23514	-0.51157	0.01023	Pass
4	-0.34767	0.10574	0.05518	Pass
5	-0.66499	0.10964	-0.38822	Pass
6	-0.64519	-0.68777	-0.56855	Pass
7	-0.15842	0.02738	-0.65325	Pass
8	-0.17031	-0.15589	0.49026	Pass
9	0.42376	-0.10569	-0.09358	Pass
10	0.10468	0.22167	0.21673	Pass

Impulse Noise:

Plotted Results:



Triangulation Results:

Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.59890,0.30008)	0.13796	-0.01630	Pass
2	(0.33974,0.23381)	(0.33802,0.23158)	0.21510	0.44584	Pass
3	(0.67751,0.20546)	(0.68000,0.20579)	-0.31090	-0.06738	Pass
4	(0.21401,0.41756)	(0.21612,0.41527)	-0.26378	0.45906	Pass
5	(0.28929,0.1877)	(0.29095,0.18694)	-0.20741	0.15297	Pass
6	(0.55774,0.49462)	(0.56190,0.50015)	-0.51930	-1.10581	Pass
7	(0.60302,0.11883)	(0.60151,0.12436)	0.18868	-1.10629	Pass
8	(0.75757,0.26965)	(0.76036,0.26477)	-0.34943	0.97607	Pass
9	(0.72261,0.29596)	(0.72082,0.29770)	0.22318	-0.34799	Pass
10	(0.05471,0.06519)	(0.05454,0.06388)	0.02185	0.26207	Pass

TDOA Results:

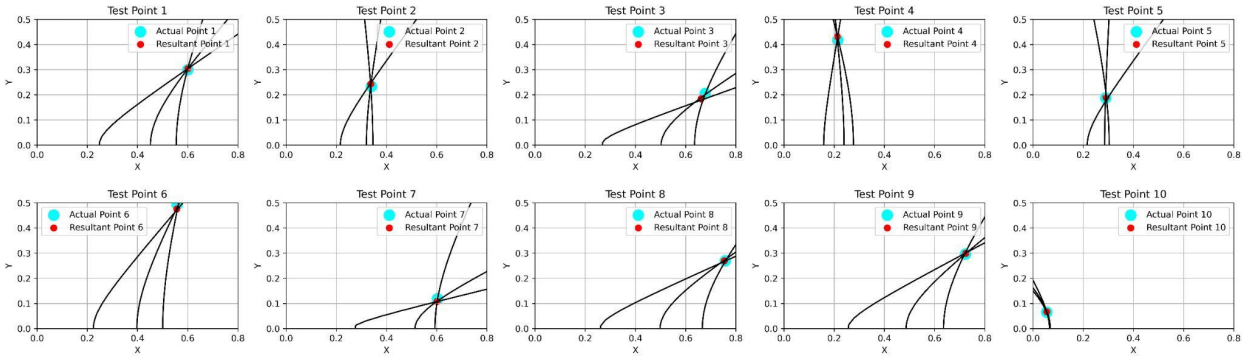
Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.15183	0.20539	-0.02569	Pass
2	0.45632	0.45549	0.42511	Pass
3	-0.23514	-0.51157	0.01023	Pass

4	-0.34767	0.10574	0.05518	Pass
5	-0.66499	0.10964	-0.38822	Pass
6	-0.64519	-0.68777	-0.56855	Pass
7	-0.15842	0.02738	-0.65325	Pass
8	-0.17031	-0.15589	0.49026	Pass
9	0.42376	-0.10569	-0.09358	Pass
10	0.10468	0.22167	0.21673	Pass

MICROPHONES IN LINE

No Noise

Plotted Results:



Triangulation Results:

Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.60025,0.30416)	-0.03081	-0.83227	Pass
2	(0.33974,0.23381)	(0.33746,0.24338)	0.28514	-1.91394	Pass
3	(0.67751,0.20546)	(0.66126,0.18426)	2.03146	4.23901	Fail
4	(0.21401,0.41756)	(0.21204,0.43230)	0.24674	-2.94784	Fail
5	(0.28929,0.1877)	(0.29052,0.18971)	-0.15366	-0.40092	Pass

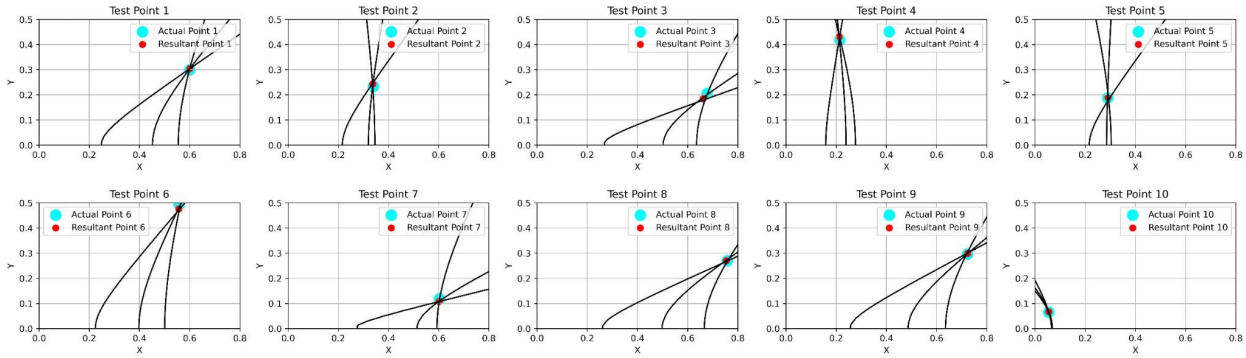
6	(0.55774,0.49462)	(0.55639,0.47601)	0.16919	3.72180	Fail
7	(0.60302,0.11883)	(0.60119,0.10798)	0.22902	2.17018	Fail
8	(0.75757,0.26965)	(0.75388,0.26975)	0.46166	-0.01986	Pass
9	(0.72261,0.29596)	(0.72351,0.29937)	-0.11321	-0.68069	Pass
10	(0.05471,0.06519)	(0.05446,0.06699)	0.03097	-0.36186	Pass

TDOA Results:

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.09897	0.21232	0.15183	Pass
2	0.67155	0.44821	0.45632	Pass
3	-0.34085	-0.26843	-0.23514	Pass
4	0.16936	0.13224	-0.34767	Pass
5	-0.63887	0.08353	-0.66499	Pass
6	-0.46387	-0.45556	-0.64519	Pass
7	-0.15874	-0.67495	-0.15842	Pass
8	-0.0121	0.22487	-0.17031	Pass
9	0.078	0.07728	0.42376	Pass
10	-0.09192	-0.32572	0.10468	Pass

Pink Noise

Plotted Results:



Triangulation Results:

Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.60025,0.30416)	-0.03081	-0.83227	Pass
2	(0.33974,0.23381)	(0.33746,0.24338)	0.28514	-1.91394	Pass
3	(0.67751,0.20546)	(0.66126,0.18426)	2.03146	4.23901	Pass
4	(0.21401,0.41756)	(0.21204,0.43230)	0.24674	-2.94784	Pass
5	(0.28929,0.1877)	(0.29052,0.18971)	-0.15366	-0.40092	Pass
6	(0.55774,0.49462)	(0.55639,0.47601)	0.16919	3.72180	Pass
7	(0.60302,0.11883)	(0.60119,0.10798)	0.22902	2.17018	Pass
8	(0.75757,0.26965)	(0.75388,0.26975)	0.46166	-0.01986	Pass
9	(0.72261,0.29596)	(0.72351,0.29937)	-0.11321	-0.68069	Pass
10	(0.05471,0.06519)	(0.05446,0.06699)	0.03097	-0.36186	Pass

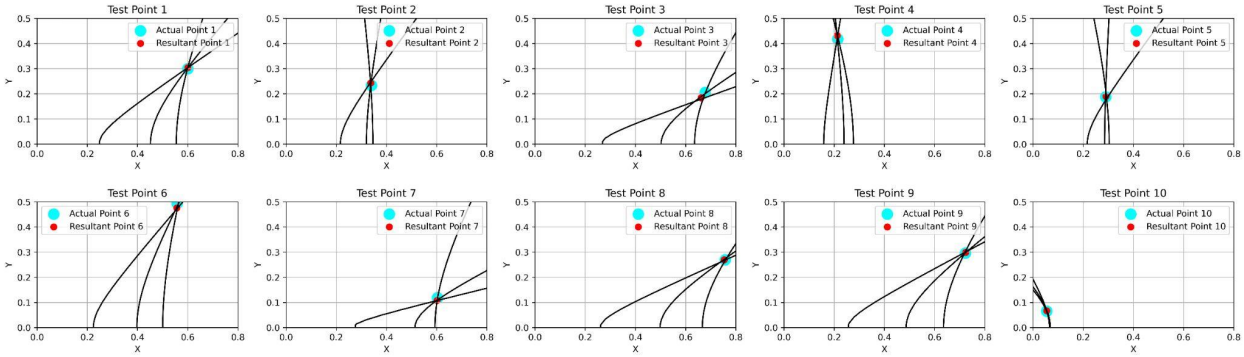
TDOA Results:

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.09897	0.21232	0.15183	Pass
2	0.67155	0.44821	0.45632	Pass
3	-0.34085	-0.26843	-0.23514	Pass

4	0.16936	0.13224	-0.34767	Pass
5	-0.63887	0.08353	-0.66499	Pass
6	-0.46387	-0.45556	-0.64519	Pass
7	-0.15874	-0.67495	-0.15842	Pass
8	-0.0121	0.22487	-0.17031	Pass
9	0.078	0.07728	0.42376	Pass
10	-0.09192	-0.32572	0.10468	Pass

Gaussian Noise

Plotted Results:



Triangulation Results:

Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.60025,0.30416)	-0.03081	-0.83227	Pass
2	(0.33974,0.23381)	(0.33746,0.24338)	0.28514	-1.91394	Pass
3	(0.67751,0.20546)	(0.66126,0.18426)	2.03146	4.23901	Pass
4	(0.21401,0.41756)	(0.21204,0.43230)	0.24674	-2.94784	Pass
5	(0.28929,0.1877)	(0.29052,0.18971)	-0.15366	-0.40092	Pass
6	(0.55774,0.49462)	(0.55639,0.47601)	0.16919	3.72180	Pass
7	(0.60302,0.11883)	(0.60119,0.10798)	0.22902	2.17018	Pass

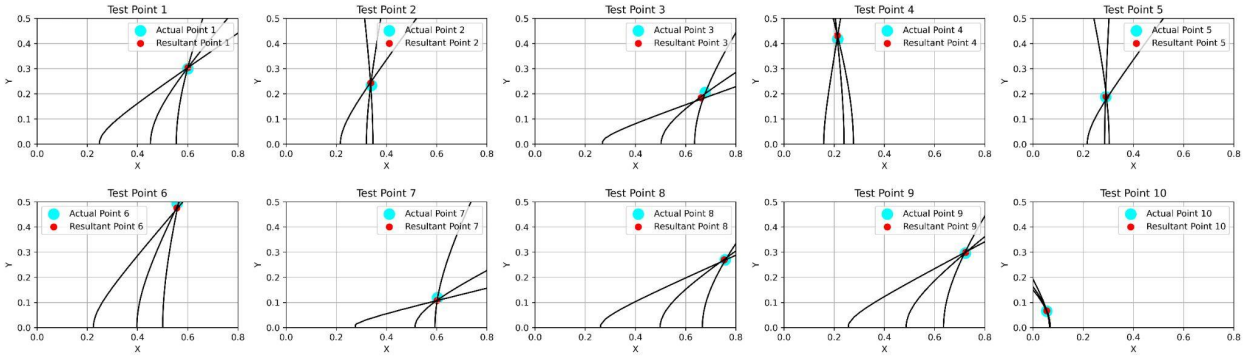
8	(0.75757,0.26965)	(0.75388,0.26975)	0.46166	-0.01986	Pass
9	(0.72261,0.29596)	(0.72351,0.29937)	-0.11321	-0.68069	Pass
10	(0.05471,0.06519)	(0.05446,0.06699)	0.03097	-0.36186	Pass

TDOA Results:

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.09897	0.21232	0.15183	Pass
2	0.67155	0.44821	0.45632	Pass
3	-0.34085	-0.26843	-0.23514	Pass
4	0.16936	0.13224	-0.34767	Pass
5	-0.63887	0.08353	-0.66499	Pass
6	-0.46387	-0.45556	-0.64519	Pass
7	-0.15874	-0.67495	-0.15842	Pass
8	-0.0121	0.22487	-0.17031	Pass
9	0.078	0.07728	0.42376	Pass
10	-0.09192	-0.32572	0.10468	Pass

<u>Impulse Noise</u>

Plotted Results:



Points	Actual Co-Ord	Estimated Co-Ord	X % Error	Y % Error	Pass/Fail
1	(0.6,0.3)	(0.60025,0.30416)	-0.03081	-0.83227	Pass
2	(0.33974,0.23381)	(0.33746,0.24338)	0.28514	-1.91394	Pass
3	(0.67751,0.20546)	(0.66126,0.18426)	2.03146	4.23901	Pass
4	(0.21401,0.41756)	(0.21204,0.43230)	0.24674	-2.94784	Pass
5	(0.28929,0.1877)	(0.29052,0.18971)	-0.15366	-0.40092	Pass
6	(0.55774,0.49462)	(0.55639,0.47601)	0.16919	3.72180	Pass
7	(0.60302,0.11883)	(0.60119,0.10798)	0.22902	2.17018	Pass
8	(0.75757,0.26965)	(0.75388,0.26975)	0.46166	-0.01986	Pass
9	(0.72261,0.29596)	(0.72351,0.29937)	-0.11321	-0.68069	Pass
10	(0.05471,0.06519)	(0.05446,0.06699)	0.03097	-0.36186	Pass

Points	Mic Pair 1 % Error	Mic Pair 2 % Error	Mic Pair 3 % Error	Pass/Fail
1	0.09897	0.21232	0.15183	Pass
2	0.67155	0.44821	0.45632	Pass
3	-0.34085	-0.26843	-0.23514	Pass
4	0.16936	0.13224	-0.34767	Pass
5	-0.63887	0.08353	-0.66499	Pass
6	-0.46387	-0.45556	-0.64519	Pass
7	-0.15874	-0.67495	-0.15842	Pass
8	-0.0121	0.22487	-0.17031	Pass
9	0.078	0.07728	0.42376	Pass
10	-0.09192	-0.32572	0.10468	Pass

ANALYSIS

NOISY VS NO-NOISE SIMULATIONS

The simulated numerical results are exactly the same regardless of the type of noise applied in the TDOA estimations, and consequently in the triangulation calculations as well. In some cases the pass/fail result is different for the noisy simulations than it is for the no-noise simulation, but this is because the criteria for no-noise situations were more stringent than the criteria for noisy situations.

Each microphone had a different noise applied using simulated noise generation. The simulated noise is generated using randomisation algorithms which typically have very low rates of correlation, as such, the simulated noise on each microphone should be almost entirely uncorrelated.

Hence, GCC-PHAT, which only picks up on similarities between correlated signals, did not change its output values.

This shows that the GCC-PHAT algorithm works as expected for uncorrelated noise. However, this test for uncorrelated noise may not translate perfectly to physical execution where certain aspects of noise can have some correlation. For example - nearby sound sources such as people speaking, or operating machinery, may result in noise readings on the microphones (which will all be in relatively close proximity to the noise source) that do have some correlation. This will be accounted for in the physical hardware testing, which is described in the section “next steps”, below.

GCC-PHAT (TDOA)

Many of the GCC-PHAT (TDOA) results pass according to the previously defined acceptable test criteria (all TDOAs within 1% of theoretical “correct” results for no-noise situations, and within 3% for noisy situations).

Testing showed that the GCC-PHAT algorithm can sometimes produce results that are completely incorrect, resulting in unpredictable estimated positions when the TDOA is passed to the triangulation algorithm. However, these errors are typically easily identifiable, and can be discarded, which is described in sections below.

MICROPHONE POSITIONING

Both configurations of microphone positions are vulnerable to the situation where the TDOA calculation fails completely and produces radically incorrect results.

However, ignoring these failed TDOA results, inspection of the results shows that placing the microphones on the four corners of the grid typically produces more accurate results, as seen by the increased number of failed tests for the in-line configuration.

This shows that the in-line configuration is superior from an accuracy standpoint.

TRIANGULATION

As per the previously defined acceptable test criteria for triangulation (x- and y- coordinates within 2% error of the actual position), the final result of the triangulation is fairly reliable.

There are two situations in which the triangulation algorithm fails:

- Small errors that fall outside of the specified acceptable test criteria, but still appear to be approximately correct
- Large errors that result in radically incorrect triangulation results

The first error can partially be attributed to overly-stringent pass/fail criteria, which can be adjusted to more reasonable values going forward. Furthermore, even if it is decided that the original criteria are adequate, the triangulation still provides a high level of reliability even within the originally defined pass/fail criteria, especially with the 4-corners mic arrangement.

Analysis of the specific results for the second type of triangulation error (large errors) shows that the root cause of these is always actually incorrect TDOA outputs. This sometimes results in situations where the triangulation algorithm fails to find an intersection between the generated hyperbolas, or produces unexpected values (e.g., values that fall well outside of the grid, or complex values). However, as with the TDOA results, these results are typically easy to identify, and can thus be discarded before they are used for the final implementation.

CHALLENGES AND LIMITATIONS AND HOW THEY WERE ADDRESSED

MULTIPLE INTERSECTIONS AND DISCARDING RESULTS

Two key problems were found with the results in general, which required some filtering and discarding of results:

- Instances where triangulation found multiple possible intersections of hyperbolas, some of which were unrealistic (typically outside of the grid). This was particularly relevant to the in-line microphone arrangement, which found results "below the grid", due to the symmetry of the configuration about the x-axis.
- Instances where triangulation found results that were clearly incorrect (e.g., results that were well outside of the grid, or complex-valued)

The triangulation algorithm found intersections between 3 hyperbolas, and took the average of these intersections to find the final estimated position result. Each intersection was individually analysed to be kept/discarded before the average was found.

Initially, the approach used was to discard all results that were complex-valued or anywhere outside of the grid. However, this ran into a key issue when one of the three intersections was slightly outside the grid due to normal inaccuracies from the simulation process, and was unnecessarily discarded. Thus, the following approach was used instead:

- Complex-valued intersections were discarded.
- Any result inside the grid was treated as valid. If multiple intersections were found within the grid, the first one was used.
- If no valid results were found inside the grid, the triangulation algorithm searches for the "most correct" result, by scanning through any identified intersections and using the one that was closest to the boundaries of the grid.
- If no valid intersections were found, the test was treated as a failure, and the result was automatically placed at the point (-10, -10)

However, this approach is still imperfect:

- Intersections that are clearly invalid by being well outside of the grid can still be identified as the "most correct" solution
- The algorithm can still identify multiple intersections within the grid, without a method to determine which one is the correct one to use.

A more comprehensive method to discard results can still be used, as described in "next steps", below

SIGNAL GENERATION AND GCC-PHAT ALGORITHM

Initially, computer-generated phase-shifted sine waves were used as the simulated signals. However, this seemed to generate extremely unreliable results from the GCC-PHAT algorithm. Multiple implementations of GCC-PHAT were tested, all producing equally unreliable results.

Ultimately, testing revealed that GCC-PHAT worked more reliably on a time-delayed version of a sound file of a real recording. Thus, the approach to signal generation was changed to use this instead, at which point a functional GCC-PHAT algorithm was easily found.

EVALUATION (ATPS)

TEST CONDITIONS:

10 Points for each test variation is conducted. Pass for overall condition is given when 8 of 10 points produce acceptable results.

MIC AT 4 CORNERS			
NOISE CONDITIONS	ATP	PASS/FAIL	COMMENTS
None	TDOA	10/10 - PASS	All values well below 1% error.
	Estimated Position	10/10 - PASS	All values well below 2% error.
Pink	TDOA	10/10 - PASS	All values well below 1% error.
	Estimated Position	10/10 - PASS	All values well below 5% error.
Gaussian	TDOA	10/10 - PASS	All values well below 3% error.
	Estimated Position	10/10 - PASS	All values well below 5% error.
Impulse	TDOA	10/10 - PASS	All values well below 3% error.
	Estimated Position	10/10 - PASS	All values well below 5% error.

MIC IN LINE			
NOISE CONDITIONS	FAILED ATP	PASS/FAIL	COMMENTS
None	TDOA	10/10 - PASS	All values well below 1% error.
	Estimated Position	6/10 - PASS	4 failed values, ranging from 2% to 5% error.
Pink	TDOA	10/10 - PASS	All values well below 3% error.
	Estimated Position	10/10 - PASS	All values well below 5% error.
Gaussian	TDOA	10/10 - PASS	All values well below 3% error.
	Estimated Position	10/10 - PASS	All values well below 5% error.
Impulse	TDOA	10/10 - PASS	All values well below 3% error.
	Estimated Position	10/10 - PASS	All values well below 5% error.

Recommended Improvements to Tests:

- More mic array positions tested to determine a better configuration or to show potential bugs and/or failing points.
- Adding more realistic noise conditions to further test robustness such as: nonlinear noise, correlated background noise, strong harmonic interference and general lower SNR conditions.
- Fluctuating volume and increasing complexity of the sound signals at the mics, such as using music or general speaking.
- Varying signal length and sample rate to find the optimal, HIGH accuracy, LOW consumption configuration.
- Consider testing all grid points to produce uncertainty maps to find problem areas

Overall Remarks on ATPs:

The simulations showed very satisfactory results. Especially with regards to noise as all results were within acceptable performance parameters. The Inline microphone configuration showed significantly more error than the Four corners configuration, even failing in the best case scenario. It would be sensible to continue with the Four corners configuration for future tests and further implementation.

NEXT STEPS

DISCARDING OF RESULTS

Simulations showed that there is a real possibility of the algorithms sometimes producing invalid results. The simulation included some methods to discard clearly incorrect results, but these still had some issues, as described in the "challenges and limitations" section. Going forward, investigation into the following approaches can be used for more reliable discarding of results:

- Discarding TDOAs that are too large to be physically possible within the confines of the grid before these are ever sent to the triangulation algorithm
- Since the triangulation algorithm finds the intersections of three parabolas and takes an average, these intersections can be compared to each other before the average is found. For example - if two intersections are within 5% error compared to each other, and the third intersection is significantly further away, the third intersection can be discarded, and the result can be calculated from just the first two intersections
- Final results that are too far away from the grid (e.g., more than 5% of the grid size away from the grid) can be discarded.

TIMING OF ALGORITHMS

Reasonable live tracking of position requires algorithms to work within a certain speed threshold. This aspect of the algorithms was not tested in the simulations, and will have to be assessed on the hardware available for the final implementation to determine the viability of live-tracking within the identified acceptable performance criteria.

Furthermore, investigation can be done into speeding up algorithms by using different hardware, or software packages (for example, there may be coding languages that can implement aspects of the triangulation algorithm faster than python).

ANALYSIS OF FURTHER SUBSYSTEMS

Simulations primarily analysed the TDOA estimation and triangulation subsystems. Some simplified versions of other subsystems (e.g., mic positioning for hardware subsystem) were also analysed. However, key implementations of other subsystems (signal acquisition, user interface, pi synchronisation, other aspects of hardware implementation) were not tested, and will need to be individually tested ahead of combination for the final implementation.

CONCLUSIONS

The following key conclusions were drawn from the simulation process:

The proposed methods of time-delay estimation can produce functional results, and are viable solutions to continue to investigate going forward.

The positioning of mics in the four corners of the grid produced the better results and will be adopted as the current “best” microphone layout.

In some instances, the methods are not able to meet the previously defined acceptable test criteria 100% of the time. However, this is largely because the acceptable performance criteria were too stringent. Particularly with the TDOA estimation, the acceptable performance criteria will be adjusted, on the basis that a larger degree of error in TDOA estimation can still produce acceptable results after triangulation.

It is evident that the methods used can sometimes produce completely inaccurate results, usually owing to failure of the GCC-PHAT algorithm in finding the correct TDOA. The final implementation will have to include rigorous processes to assess the validity of results and discard invalid results. It will also be helpful to include processes that can identify failed calculations and report them to the user. Finally, it will be useful to assess the rate of failure on the final hardware implementation, so that this can be made evident to the user.

REFERENCES:

[1] Free Audio Zone, “Finger Snap Sound Effect,” YouTube. May 08, 2020. Accessed: Sep. 15, 2023. [Online]. Available: https://www.youtube.com/watch?v=C-VpXmkk0mw

[2] Y. Xiong, “GCC-PhaT Library,” GitHub, Mar. 24, 2017. https://github.com/xiongyihui/tdoa/blob/master/gcc_phat.py (accessed Sep. 15, 2023).

[3] Ritu and S. K. Dhull, “A comparison of Generalized Cross-Correlation methods for time delay estimation,” Social Science Research Network, Nov. 2017, [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3072083

[4] C. Knapp and G. Carter, “The generalized correlation method for estimation of time delay,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 4, pp. 320–327, Aug. 1976, doi: 10.1109/tassp.1976.1162830.

[5] X. Wen and J. Wang, “TDOA Location Accuracy Experiment,” Journal of Physics, vol. 1237, no. 3, p. 032031, Jun. 2019, doi: 10.1088/1742-6596/1237/3/032031.

[6] “Object Tracking using Time Difference of Arrival (TDOA) - MATLAB & Simulink - MathWorks United Kingdom.” https://uk.mathworks.com/help/fusion/ug/object-tracking-using-time-difference-of-arrival.html