Real Time Sound Source Localization

By

PC Arslan Moveed

PC Muhammad Ali Farooq

ASC Muhammad Aqib Niaz

Submitted to the Faculty of Department of Electrical Engineering, Military College of Signals, National University of Sciences and Technology, Islamabad
In partial fulfillment for the requirements of a B.E Degree in Telecom Engineering

May 2015

Abstract

This report presents a method for localizing a target based on the sound it produces. The project is implemented in order to locate a target not visible to the naked eye, using unconventional means, like radar. This project would be a cheap alternative to conventional localization techniques. The source is localized using the time delay of arrival technique. The time delay of arrival technique calculates the delay of arrival of sound on to multiple microphones on an array and derives an angle from the delay calculation. The hardware consists of an array of two microphone and three USB cameras, where each camera covers 60 degrees of the 180 degrees field of view. The hardware detects the direction of the source in real time and displays the output on a compass as well as showing the real time video stream of the target's location on a GUI. The project is intended to locate a gunshot or sniper in real time in the battle field. The hardware was tested using intense sounds to replicate the gunshots as real gunshots could not be used. The hardware worked marvelously in noisy and reverberant environments. The angles calculated lay between +2 and -2 of the actual angle.

It is hereby certified that the contents and form of the project report entitled “Sound Source Localization”, submitted by the syndicate of

1. PC Arslan Moveed
2. PC Muhammad Ali Farooq
3. ASC Muhammad Aqib Niaz

has been found satisfactory as per the requirement of the B.E. Degree in Electrical (Telecom) Engineering.

Supervisor:

Lt Col Dr. Muhammad Tayyab Ali

R&D wing,

MCS, NUST

DECLARATION

We hereby declare that no content of work presented in this thesis has been submitted in support of another award of qualification or degree either in this institution or anywhere else.

DEDICATED TO

Almighty Allah,

Faculty for their help

And our parents for their support

ACKNOWLEDGEMENT

We are thankful to Allah Almighty for giving us knowledge, strength and perseverance to accomplish this task successfully. For nothing in this world happens without the will of Allah.

We would especially like to thank our supervisor Lt Col Dr Muhammad Tayyab for his support and understanding demeanor throughout the course of this project. We would also like to thank Ms Maryam Rasool (EE Dept) for her utmost patience and support through trying times.

We would also like to thank the people from NDC and DESTO, who provided assistance in many difficult matters.

Last but not least, we are very thankful to our parents, who bore with us in times of difficulty and hardship. Without their consistent support and encouragement we could not have accomplished our targets successfully.

Table of Contents

Contents

List of Abbreviations

TDOA	Time Delay of Arrival
AOA	Angle of Arrival
GCC	Generalized Cross Correlation
TDE	Time Delay of Arrival
LCD	Liquid Crystal Display
MATlab	Matrix Laboratory
USB	Universal Serial Bus
PnP	Plug and Play

List of figures

1. Chapter 1

1.1. Introduction

1.1.1. Background

The acoustic source localization is an important and relatively new technique for determining the location of a particular target using its acoustic signature [4]. The project forms the basis a passive localization system, unlike a radar, for locating events which produce noise, e.g. gunshots, bomb blasts, speech of a person etc.

This project was implemented to achieve a means for locating a target source and its potential applications for locating a sniper in an open battle field scenario as well as other defense and civil applications.

1.1.2. Problem statement

"Design and implement a real time acoustic source localization system prototype"

1.2. Project description

The sound source localization system consists of three parts, a sensor, in our case, microphones, a processing unit, in our case a laptop, and an out display, an LCD. The location of a target is estimated using the method of generalized cross correlation in frequency domain. The sound from the target reaches the microphones at a speed of 329 ms^{-1}. If more than one microphones are used and placed spatially apart, the sound from a single source reaches both the microphones at different times, depending upon the location of the sound producing source and the microphones. To determine the angle of arrival of the sound, relative to the microphone array, we first need to calculate the time delay inherent in the received signals. This delay is then transformed into the bearing angle of the target source. To calculate the time delay, the method of generalized cross correlation is used. In this project the location of a target source is being determined by calculating the angle of the source relative to the microphone array. The calculated angle is then being displayed on a GUI and a real time video stream is being displayed on the LCD monitor.

The acoustic sensors are microphones, which receive the sound from the source and relay it to the laptop. The MATlab processes the signals from the microphones to get the source bearings and display the result in a GUI.

1.3. Prospective Application Areas

1. Defense and military applications to locate artillery and snipers etc.
2. Police and law enforcement agencies, to enable accurate and rapid reaction
3. Domestically, in buildings and big properties to locate potential break-ins
4. Conference rooms,
5. Telepresence systems
6. This project will add a new surveillance technique to better locate targets.

1.4. Scope, Objectives, Specifications and Deliverables:

1.4.1. Scope and Objectives:

The scope of this project was to achieve the following objectives,

1. Designing a passive localization system
2. Understanding the behavior of sound in noisy and reverberant environments
3. Designing an efficient algorithm for calculating the time delay inherent in sounds received from different microphones
4. Designing the microphone to PC interface for connecting the microphones to sound card of the laptop
5. Implementing the algorithm on MATlab

Following are the goals we achieved with the project

1. Efficiently determining the location of target source in horizontal plain
2. Modular design , we can replace the microphones without making any major change in the setup
3. Wide variety of targets may be localized, e.g. gunshots, claps, voice etc.

1.4.2. Specifications:

1. LM358 amplifier microphones.
2. 3 x USB PnP cameras
3. Input: 2 x microphones to acquire sound
4. Recommended hardware: a PC with at least 1.6GHz CPU and 2GB memory.
5. USB hub, tripod and camera mount.

1.4.3. Deliverables

1. Microphone array
2. Camera mount
3. Tripod stand
4. Microphone interface

2. Chapter 2

2.1. Literature review:

2.1.1. Over view of existing literature:

The sound source localization system is responsible for detecting a target, based upon its acoustic signature, and displaying the results in a GUI. Following are the function s performed by the system,

1. Detect sound in the environment where the system is placed
2. Determine whether the detected sound is voiced(useful) or not
3. Calculate time delays between the signals received from multiple sensors
4. Determine angle of arrival based upon the time delays I n received signals
5. Display angle on GUI
6. Show live stream of the location of source

2.1.2. GCC

In signal processing, **cross-correlation** is a measure of similarity of two series as a function of the lag of one relative to the other [5][6]. This is also known as a sliding dot product or sliding inner-product. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology.

For continuous functions *f* and *g*, the cross-correlation is defined as**:**

$$(f \star g)(\tau) \overset{\mathrm{def}}{=} \int_{-\infty}^{\infty} f^*(t)\ g(t+\tau)\,dt,$$

Where f^* denotes the complex conjugate of f and τ is the lag.

Similarly, for discrete functions, the cross-correlation is defined as**:**

$$(f \star g)[n] \overset{\mathrm{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\ g[m+n].$$

In this section the frequency-domain implementation of GCC is briefly reviewed. First, two time-domain signals are individually transformed into the frequency-domain in which they are multiplied with each other (after taking the complex conjugate of one of them), giving $G_{left,right}$. This result is then transformed back to the time-domain to obtain a correlation function. The position corresponding to the maximum cross correlation will indicate the TDOA [6]. On its turn the TDOA will indicate the angle of sound incidence, given the array geometry. For robustness, the popular PHAT weighting scheme can be used to obtain a unity gain for all frequency components, while preserving phases which contain the actual delay information.

$$\hat{G}_{Left,Right}(f) = \frac{G_{Left,Right}(f)}{\left|G_{Left,Right}(f)\right|}$$

Theoretically, when transforming back to the time-domain, this should make the correlation function a unit impulse function (neglecting noise, echo,). As a result, additional peaks in the correlation (from echo paths and noise sources) will not influence the spike from the direct path as much, giving better location estimations. Given two signals xi(n) and xj(n) the GCC-PHAT is defined as:

$$\hat{G}_{PHAT}(f) = \frac{X_i(f)[X_j(f)]^*}{|X_i(f)[X_j(f)]^*|}$$

Where Xi(f) and Xj(f) are the Fourier transforms of the two signals and $[\ ]^*$ denotes the complex conjugate. The TDOA for these two microphones is estimated as:

$$\hat{d}_{PHAT}(i,j) = \underset{d}{argmax}\left(\hat{R}_{PHAT}(d)\right)$$

2.1.3. TDOA

The TDOA or time delay of arrival is the difference in arrival time of a particular excitation on to multiple spatially separate sensors [6]. In this case the microphones are place 30cm apart on the array [2], the sound from a source reaches the microphone at different times, depending upon the location of the sound source as well as the array relative to it. The TDOA then determines the angle of arrival or AOA. A minimum of two sensors are required for calculating the delay in a single dimension. We employ two microphones as on two are supported by the sound card.

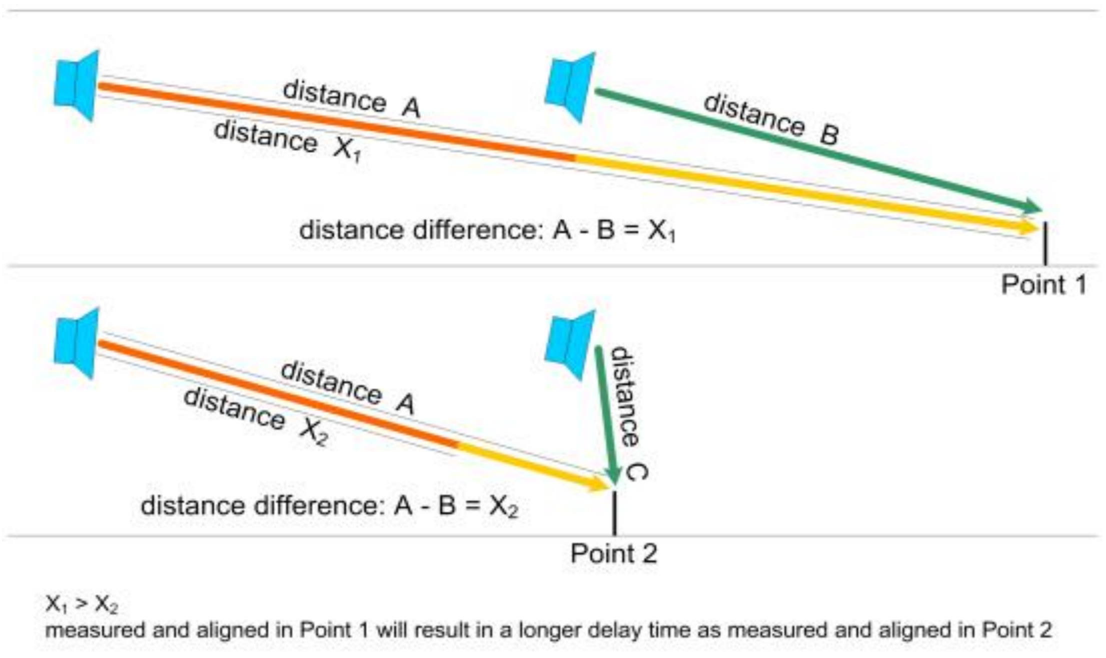

distance A
distance X_1
distance B
distance difference: A - B = X_1
Point 1
distance A
distance X_2
distance C
distance difference: A - B = X_2
Point 2
$X_1 > X_2$
measured and aligned in Point 1 will result in a longer delay time as measured and aligned in Point 2

Figure 1: TDOA

TDOA of a signal can be exploited to extract the actual angle from where the signal is emanating. One of the microphones is considered as the primary and the other, the secondary. The time TDOA is calculated relative to the primary, and depending upon its magnitude, the AOA is determined. If the source is exactly in front of the array, the TDOA is calculated as zero, in this way, towards one side we get the maximum positive TDOA, and towards the other we get the minimum possible TDOA.

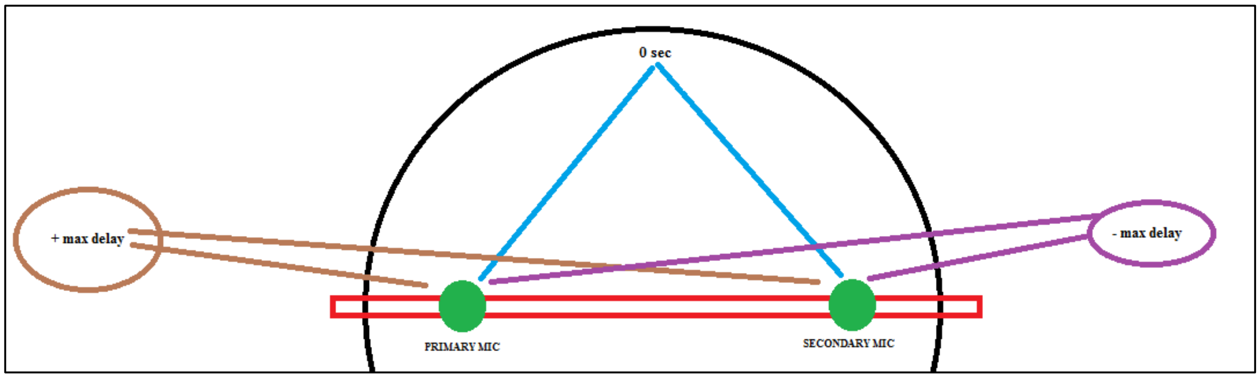


Figure 2: Time delays, relative to microphone array

The figure shows the TDOAs calculated, relative to the primary microphone.

2.1.4. AOA

AOA or the angle of arrival is calculated using the TDOA, by employing trigonometry. The AOA correctly states the location of the target source [7].

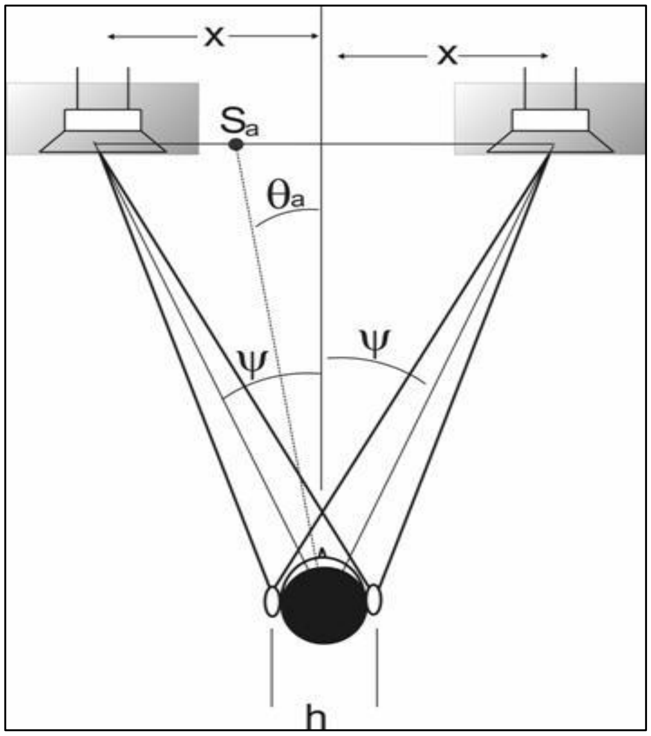


Figure 3: Angle of Arrival

The AOA is directly dependent on the TDOA calculation. The angle is zero if the source is directly in front of the array, it is +90 degrees for the +maximum delay and -90 degrees for the –maximum delay.

2.2. Problem formulation:

The sound source localization system is a relatively new surveillance and localization technique. To achieve the localization, we opted for the minimum number of sensors and effectively calculating the angle of the source relative to the microphone array in a 180 degrees field of view in front of the array. The system would detect sound through the microphone array mounted a-top a tripod stand. The detected signals are then subjected to A/D conversion by the sound card of the laptop. The digitized signals are then processed and angle of arrival is extracted from them. The results are then displayed on a GUI designed on MATlab.

3. Chapter 3

3.1. Detailed Design:

The project is divided into two major parts, the hardware design and the software development. The hardware consists of the interfacing circuitry and the sensors as well as the output displaying device i.e. the LCD monitor and the cameras. The software part consists of the actual algorithm for calculating the angle as well as the code for the GUI made in MATlab.

The sensors would send the signals that they pick up, to the laptop and the MATlab algorithm in the laptop would determine the angle of arrival based upon various factors in the detected signals.

The angle, so determined by the algorithm are then be displayed on a GUI and shown on the live video stream from one of the three cameras.

The GUI displays the angle in form of a compass and the cameras show the live stream of the area where the targets location is determined.

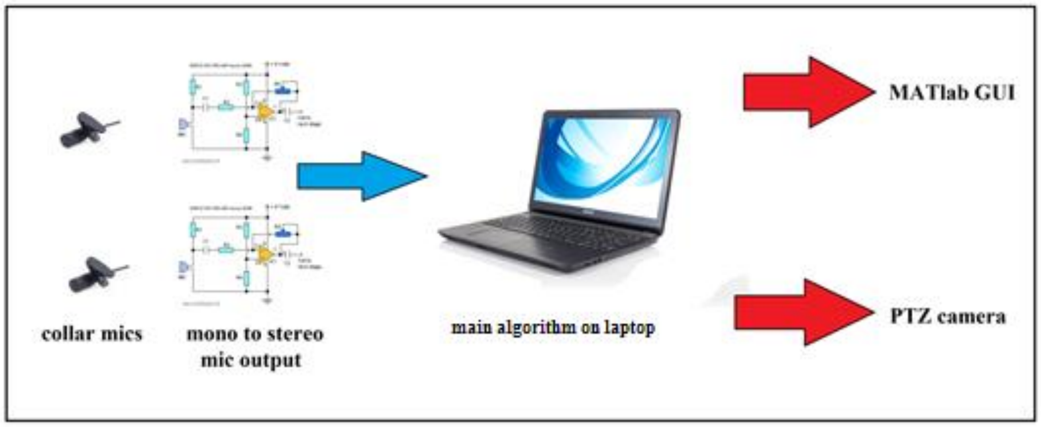


Figure 4: System Flow Design

The figure 4 illustrates the flow of the project, i.e. the sound is captured by the microphones and sent to laptop where it is processed upon. The laptop then displays the results on GUI and shows the live stream of the target.

3.2. Hardware design:

The hardware part consists mainly of the sensors and the cameras. The sensors are mounted on an array, and attached onto a tripod stand. A circular camera mount is used to mount three USB webcams, each covering 60 degrees of the total 180 degrees field of view. The camera mount is also housed with the tripod stand. The stand also houses the various circuitry for interfacing the microphones with the laptop. The details about each hardware component are illustrated below.

3.2.1. Microphones:

The sensors used for detecting the sound of the target source are the microphones. The microphones work on the same principle as speakers, only in reverse. The vibrations In the air caused by the sound source travel towards the microphones and cause the filament to move, this motion is exactly the same as the sound produced. These vibrations are in turn converted to electrical signals. These signals are perceived as sound by the sound card of the laptop.

The microphones used for the project are LM358 electret microphones. They are readily available in the market.

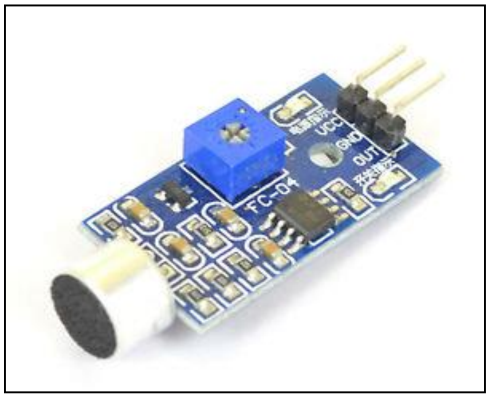


Figure 5: LM358 Microphone

The figure 5 shows the LM358 electret microphone. The circuitry consists of the LM358 amplifier and various resistors and capacitors necessary to achieve the amplification of the detected signals. A potentiometer allows the range to be adjusted. The microphones have a good dynamic range and also prove useful as the level of noise is low for highly intense sounds, which the syndicate used for testing the system. The microphone has three pins, a VCC pin, a GND pin and an OUT pin. The specifications of the pins are as under

1. VCC = 2V-32V, we used 5V power supply for the microphones
2. GND = GND is provided by the GND pin of the power supply
3. OUT = The OUT pin outputs the analogue signal which is fed into mono to stereo interface

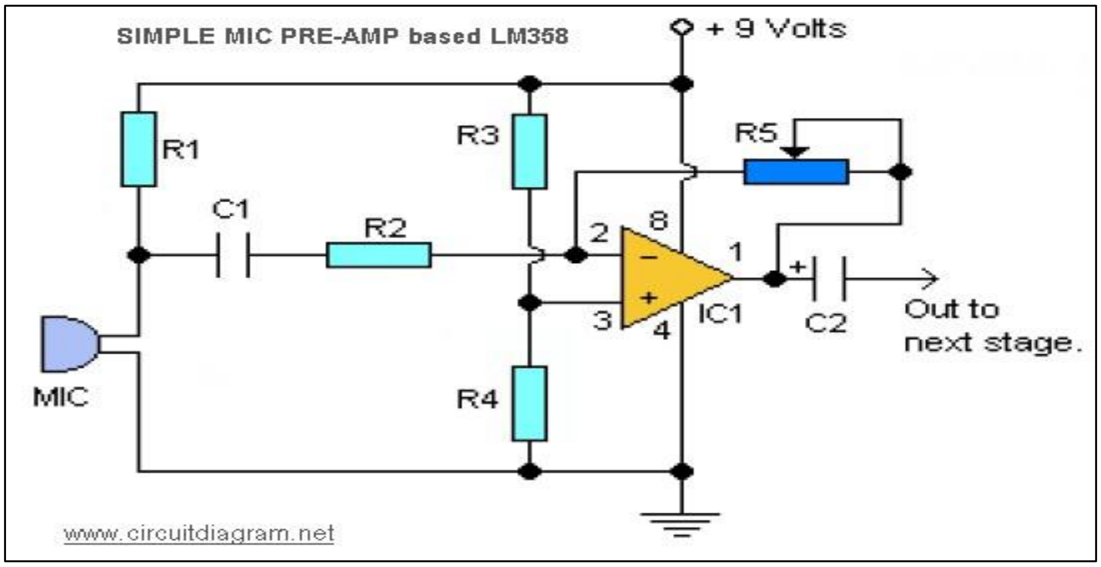


Figure 6: LM358 Microphone Amplifier Circuit

The figure 6 shows the details of the circuitry for the amplifier in the microphone.

3.2.2. Microphone array:

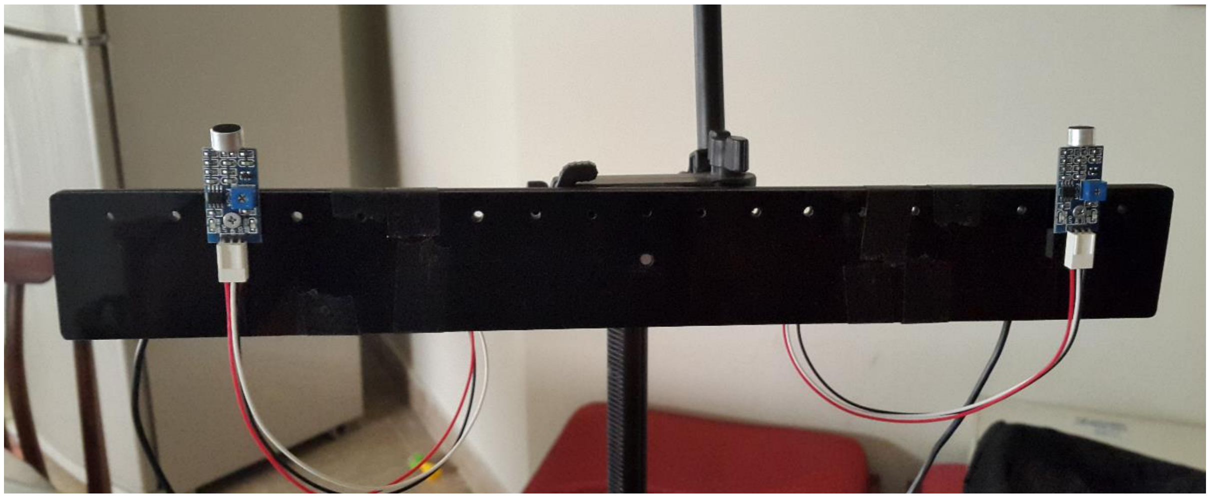


Figure 7: Microphone Array with Microphones

The figure 7 shows the two microphones mounted on the array. The microphones are mounted on to and array, the minimum number of microphones are used, i.e. two, which are supported by the sound card of the laptop and are required for determining the angle in one dimension. Each microphone outputs a mono analogue signal, which is fed into the laptop on one of the two channels of the sound card.

To get the optimum delay between the sounds captured by the two microphones, we keep the microphones 30 cm apart, this would mean that the maximum delay experienced would be given by the relation

$$S=VxT$$

Here ‘S’ is the distance between the two microphones, kept at 30cm, ‘V’ is the speed of sound in air, i.e. 340ms^{-1}, and ‘T’ is the time delay. Now, for maximum time delay, the source must be exactly perpendicular to the array, on either side, and thus the signal would travel atleast 30cm further after reaching one of the microphones. Therefore, the maximum delay experienced would be, +0.9e-3sec or -0.9e-3sec depending upon the location of source relative to the primary microphone [2].

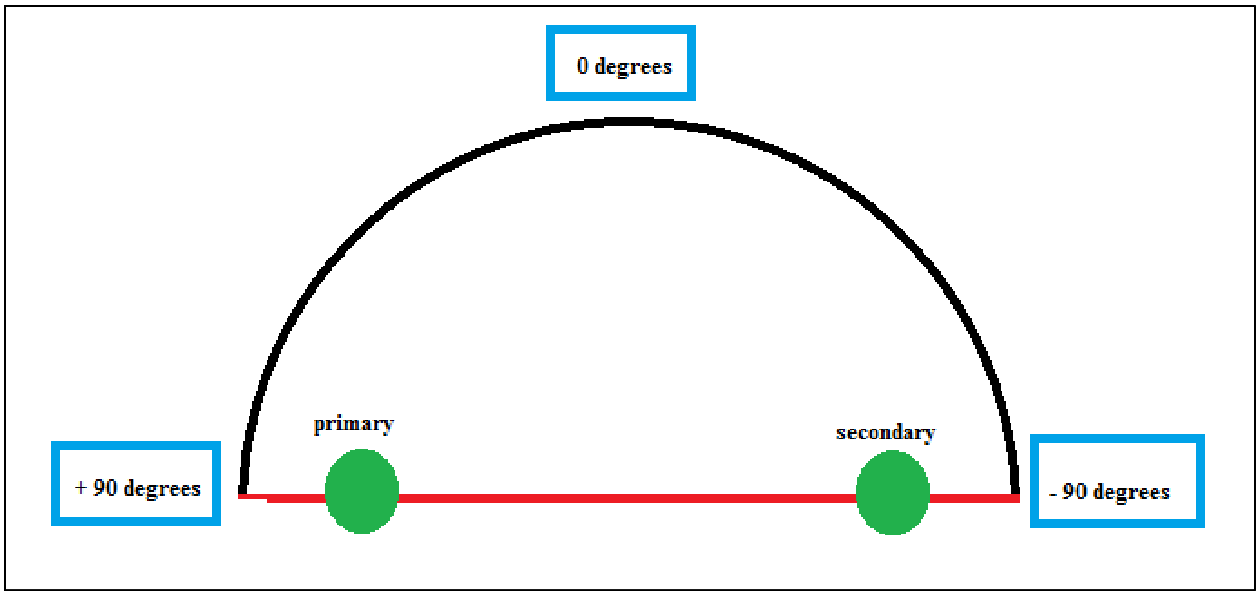


Figure 8: Angles of Arrival relative to array

The figure 8 illustrates the angle of arrivals that are calculated relative to the microphone array. The two microphones used are only able to provide the location of the target in one dimension, in our case the horizontal axis and in front of the array in a 180 degrees field of view.

3.2.3. USB PnP Cameras and camera mount:

Figure 9: USB PnP Camera

The camera shown in the figure 9 is a USB plug and play webcam with a resolution of 640x480 pixels. Three such cameras are used to provide real time video of the source of sound. These cameras are mounted on a camera mount which is circular. Each camera covers 60 degrees field of view out of the total of 180 degrees. Once the angle is correctly determined, real time video stream of the camera covering the particular sector is displayed on the video field in the MATlab GUI.

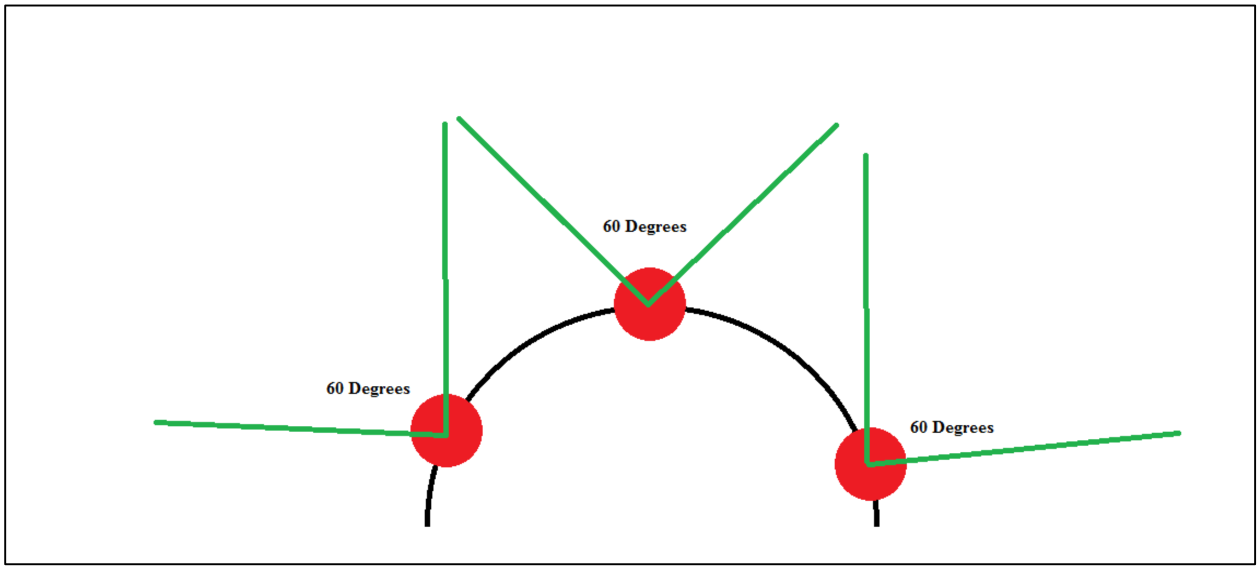


Figure 10: 60 degrees coverage by each camera

The figure 10 shows how the three cameras are used to cover 180 degrees field of view.

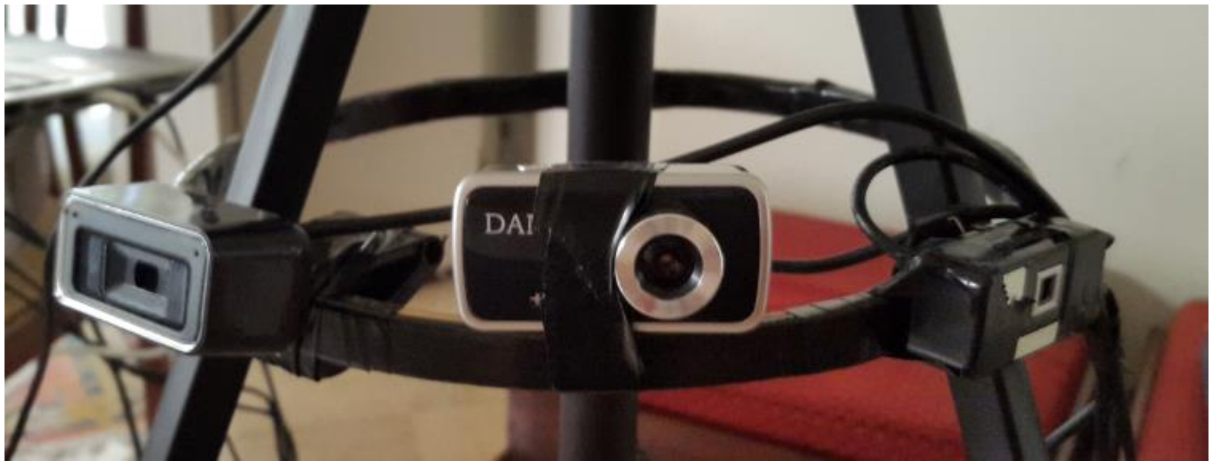


Figure 11: Actual Camera mount on tripod stand

The figure 11 shows the actual camera mount attached to the tripod stand.

3.2.4. Interfaces:

Various interfaces are used in the project, they are as follows,

1. **USB hub :**

   A USB hub is used to connect the three cameras with the laptop. The cameras are accessed by MATlab using the addresses. The USB hub supports 6x devices at a time and the cameras worked stupendously with the hub

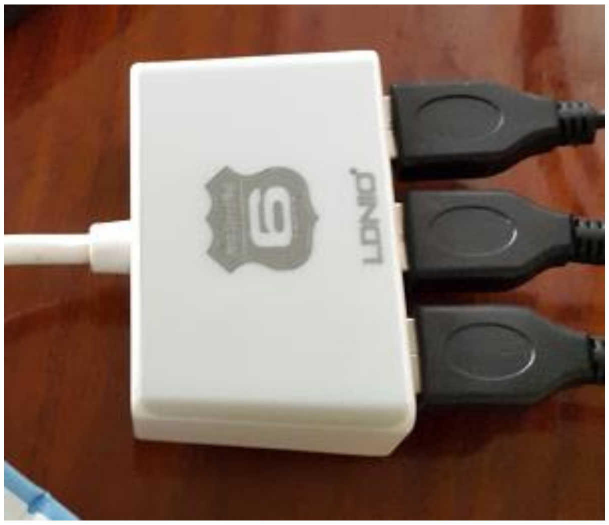


Figure 12: USB Hub

The figure 12 shows the three cameras connected to the USB hub.

2. Mono to Stereo converter:

Each microphone outputs a mono channel signal. Since two microphones are used, a mono to stereo converter circuit was designed. It serves three purposes

a. Provides 5V Vcc to the microphones
b. Provides GND terminal for the microphones
c. Relays the mono signals from each microphone on to a single stereo pin

The circuit is implemented on a Varro board. Stereo audio jacks (female) are used, and the microphones are connected to the circuit using auxiliary cables readily available in the market. The auxiliary cables are shielded and provide a means for avoiding the assimilation of airborne EM waves to an extent.

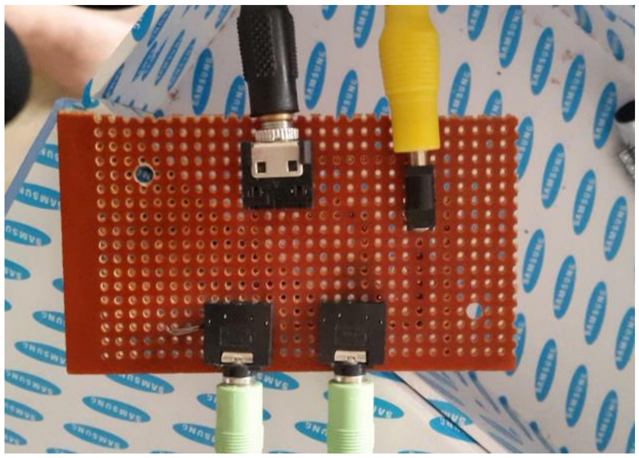


Figure 13: Mono to Stereo conversion circuit

The figure 13 shows the mono to stereo converter circuit. The green jacks are from the microphones, the yellow jack is the power supply and the black jack goes to the laptop sound card.

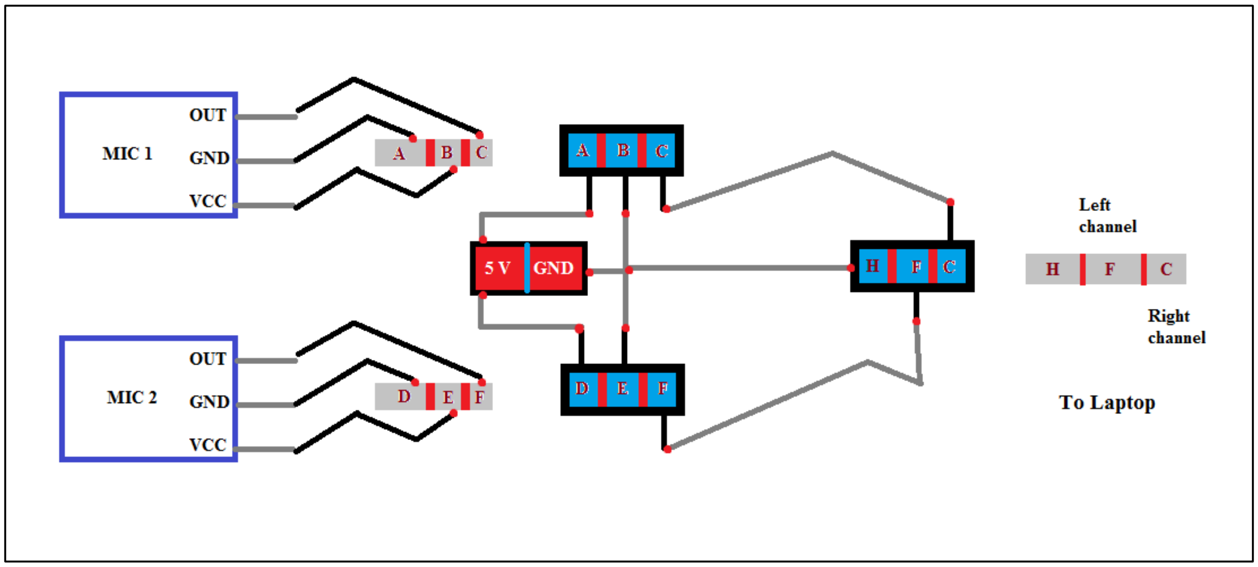


Figure 14: Circuit Diagram for Mono to stereo converter circuit

The figure 14 illustrates the circuit, how it connects the 3-pin mics with mono output to a stereo jack.

3.2.5. Tripod stand:

The whole assembly, i.e. the microphone arrays and the camera mount is mounted on to a tripod stand. The height is adjustable and this feature is needed so that the height of the microphones is high enough to avoid the reverberations caused by the floor.

Figure 15: Tripod stand

3.3. Software development:

The algorithm is implemented on MATlab. A large number of methods are available for calculating time delays. The GCC, generalized cross correlation [5] is chosen for the project. The choice of the TDOA [6] algorithm is influenced very much by the environment. The very noisy and reverberant environment causes haphazard delay calculation, which are of no use to us. Therefore a robust method is required for calculating the TDOA in noisy and reverberant environment. Hence the GCC algorithm is chosen.

The main function of the GCC algorithm is to obtain a correlation graph of the signals detected by the two sensors. This correlation graph would enable us to determine the delay between the two signals. The flow of the algorithm is shown in the figure.

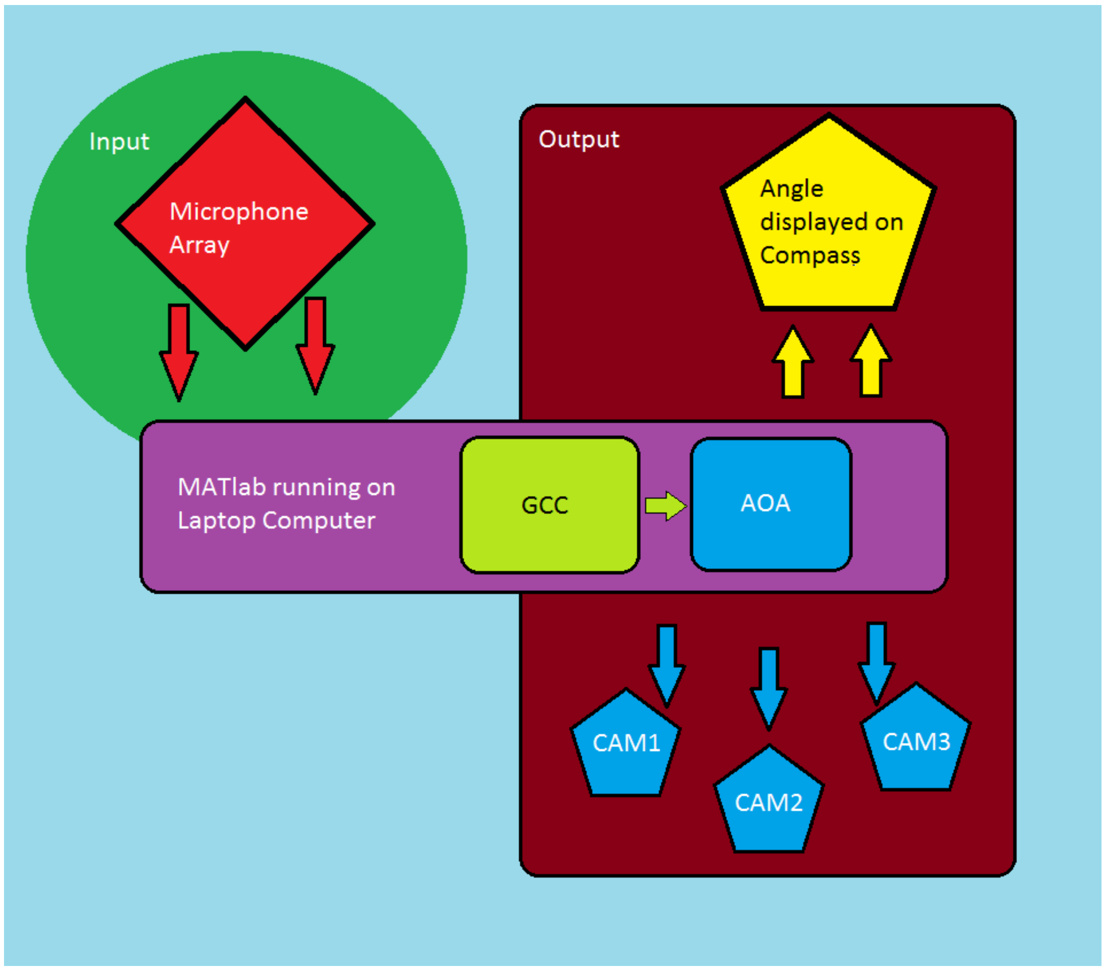


Figure 16: Software Flow Chart

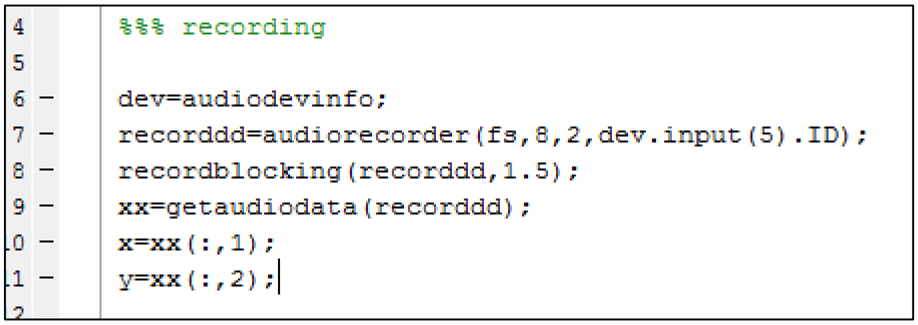
The figure 16 illustrates the following steps followed in the development of the algorithm,

1. Acquire input from microphones
2. Check whether the signals have requisite energy
3. The selected frames are sent for processing
4. The TDOA is derived from the frame, and in turn the AOA
5. Display the AOA on GUI

The microphones, once active, pick up all types of noise, EM waves, Wi-Fi, mobile cellular signals etc. There are ways to filter out most of the noise, however the signals received are not pure. Therefore the only option is to work with the received signals. The filters designed on MATlab play a huge role in cleaning the signals to some extent. Also, since the system is turned on, it has to gather intelligence about the surrounding environment. This is done in order to process only such signals that have an energy greater than a previously determined threshold value, this save both time and does away with any and all unwanted angle calculations.

3.3.1. Acquiring signals from microphones:

The signals are acquired from the microphones through the sound card of the laptop on MATlab. The data acquisition toolbox of MATlab makes it possible for acquiring the inputs. The signals read in MATlab are digitized, the parameters are set by the user, including the sampling frequency, number of bits, the type of channel from where the input is received and the address of the port from where the input is being read.



```
%%% recording

dev=audiodevinfo;
recorddd=audiorecorder(fs,8,2,dev.input(5).ID);
recordblocking(recorddd,1.5);
xx=getaudiodata(recorddd);
x=xx(:,1);
y=xx(:,2);
```

Figure 17: MATlab code excerpt

The figure 17 shows the code excerpt for recording and acquiring the sound signals from the microphones. The following commands are used,

1. Audiodevinfo : Get information about audio devices connected with the system
2. Audiorecorder : Make an audio-recorder object to store the recorded signal
3. Recordblocking : Record from the audio devices for a particular time period
4. Getaudiodata : Acquire just the amplitudes of the samples of the recorded signals

The acquired stereo signals are separated and stored in separate variables. After this the processing takes place.

3.3.2. GCC:

After preprocessing, the signals are sent to the main GCC function where the TDOA of the signals is calculated and the AOA is determined. The GCC technique is based on the assumption that the correlation of the delayed versions of the same signal is maximum at the point of the delay. Therefore, once the GCC of the two delayed signals is plotted, there is seem a distinct peak which shows the maximum correlation of the two signals.

Considering the signals from the two microphones as the delayed versions of a single signal, we can say

$$S_1 = s(t+d_1) + n_1 \qquad \text{------------------------------(i)}$$

$$S_2 = s(t+d_2) + n_2 \qquad \text{------------------------------(ii)}$$

Here, we can say see that signals s1 and s2 are delayed versions of the signal s(t) from a single sound source. N1 and N2 are the noise paths of the two microphones. The time delay to be calculated for source localization is essentially, “d_1-d_2”.

The signals from the two mics ‘i’ and ’j’ are cross correlated as follows,

$$\boxed{R_{ij}(\tau) = \sum_{n=0}^{N-1} x_i\,[n]x_j[n-\tau]}$$

This equation depicts the time domain cross correlation, however, due to noise, the time domain cross correlation gave bad results, thus cross correlation was done in frequency domain, as follows,

$$\boxed{G(f) = X_i(f)X_j(f)^*}$$

Here, Xi(f) and Xj(f) are the frequency domain representations of the two signals from mics ‘i’ and ‘j’, and Xj(f)* depicts the complex conjugate of the signal Xj. the signals in frequency domain are multiplied with each other and the product graph is called the GCC plot, depicting the correlation of the two signals.

In MATlab the algorithm is implemented in frequency domain. The signals are first converted to their respective frequency components. The signals now show their magnitudes and phases. After multiplication of one signals with the conjugate of the other, the phases are diminished and a maximum is shown in place of the sample delay, which when converted back into time domain, yields the cross correlation plot showing the sample delay.

The sample delay is converted to time delay by dividing it with the sampling frequency, in our case 44100 Hz is used. The TDOA is then converted into the AOA.

3.3.3. AOA:

The position of the target is estimated in terms of angle of arrival, or AOA. The angle is derived from the TDOA calculated from the two signals. The two microphones used, give the angle in the horizontal plain only and in front of the array. The angle is calculated by employing trigonometry.

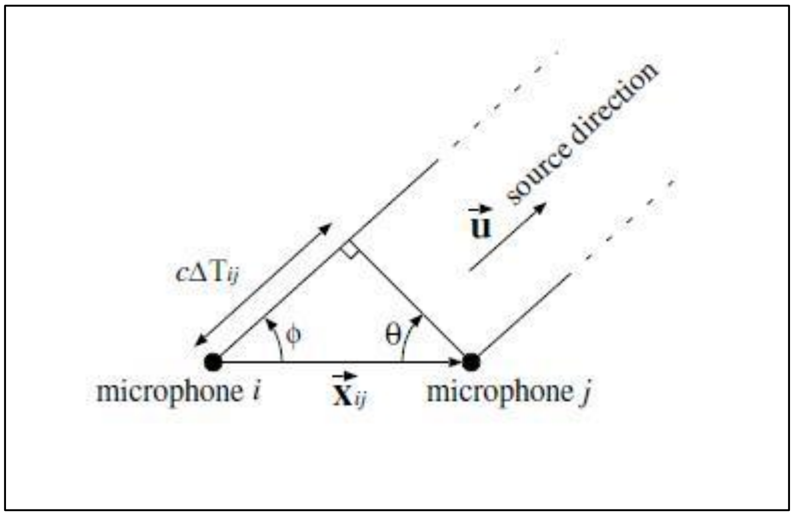


Figure 18: Angle of arrival

From the figure, the '**xij'** is known to us as the separation between the two microphones, this is the hypotenuse. The distance **'c*Tij'** is the base for the angle that is the AOA. Applying the relation,

$$\boxed{\cos\varphi = \sin\theta = \frac{c\Delta T_{ij}}{\|\vec{x}_{ij}\|}}$$

In this way the AOA is calculated. This angle is then displayed on a compass in MATlab GUI and a live video stream is also shown [7].

Chapter 4

4.1. Project Analyses and Evaluation:

The simulation results are illustrated below.

4.1.1. Acquiring signals from microphones:

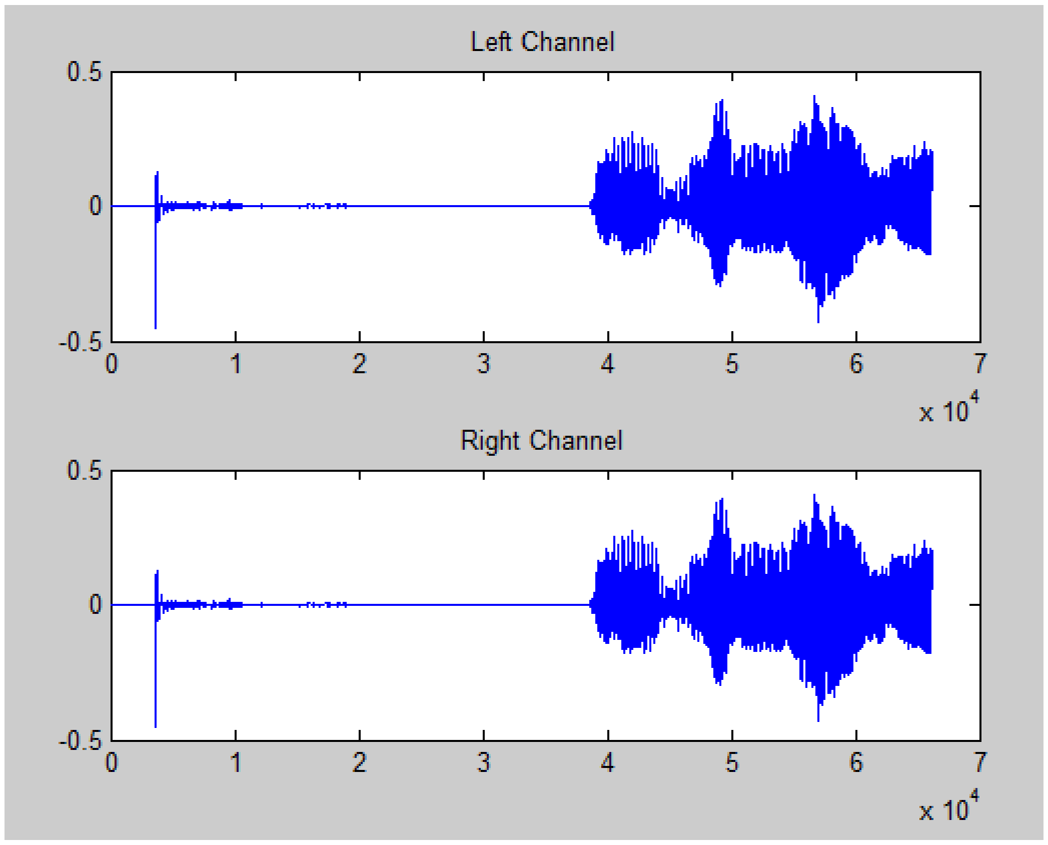


Figure 19: Acquired signals from microphones

The figure 19 shows the time domain signals captured by the microphones. The two signals from the microphones look the same, however, the change is in the time component, i.e. the signals are delay against the time. The next figure will show the delay clearly.

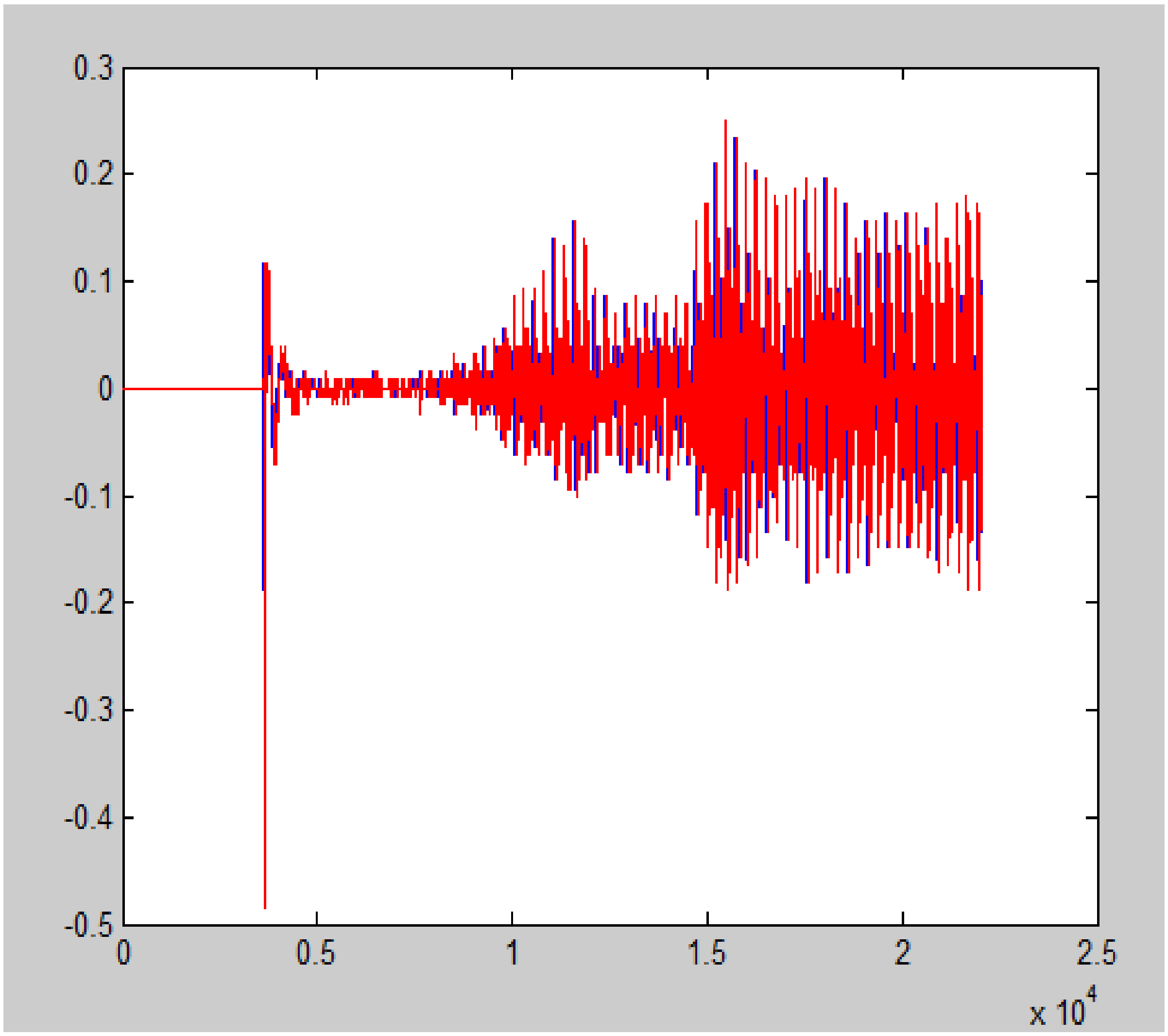


Figure 20: Overlapped signals from the two microphones

The figure 20 shows that once the signals are plot again each other, the delay is evidently seen. Further, we can zoom in and see the delay

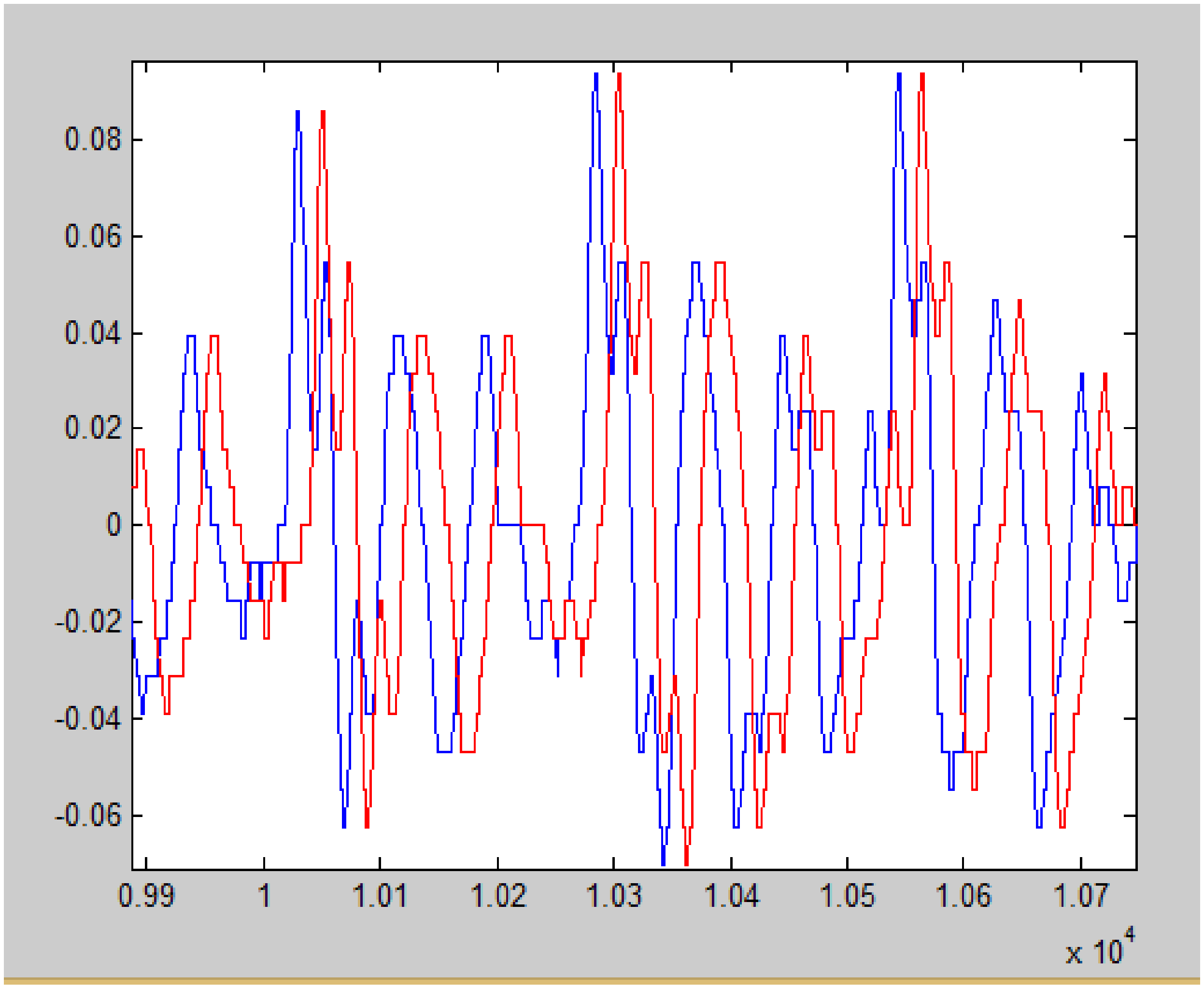


Figure 21: Time domain delayed signals

In figure 21 we see that the two signals are identical and only vary in their phase. This is of interest to us as this is the TDOA we need to calculate.

4.1.2. Filtration:

Filtration play a key role in enabling the efficient correlation. The unwanted noise is simply band-pass filtered out of the signal, and the remaining part is sent ahead for processing.

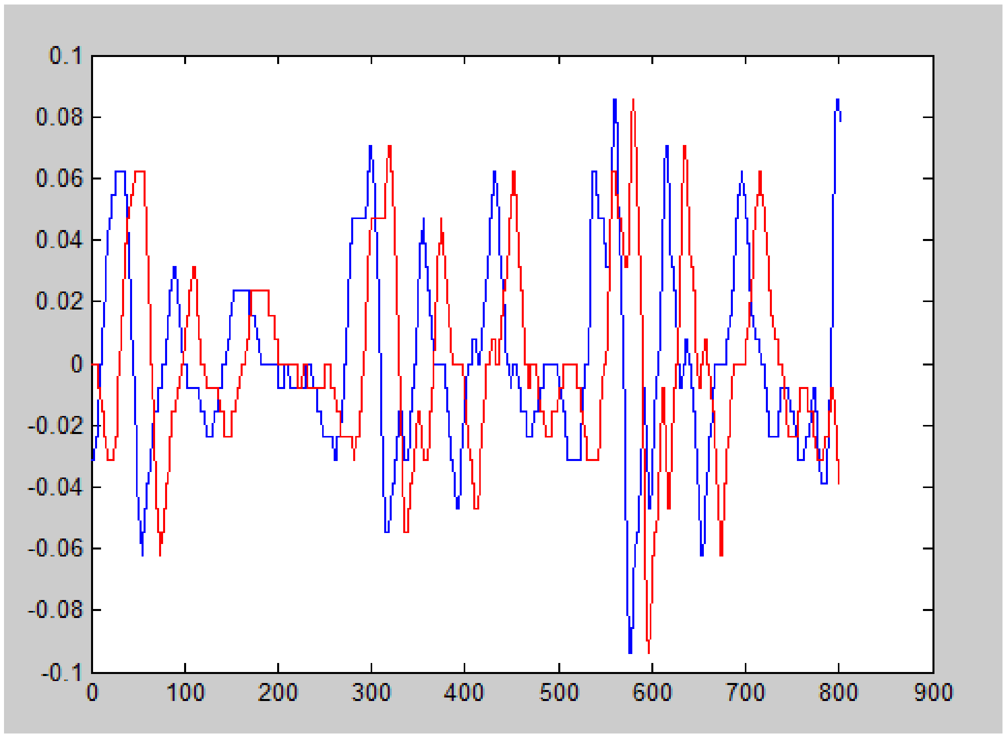


Figure 22: Actual recorded signals

The figure 22 shows the signals before being filtered.

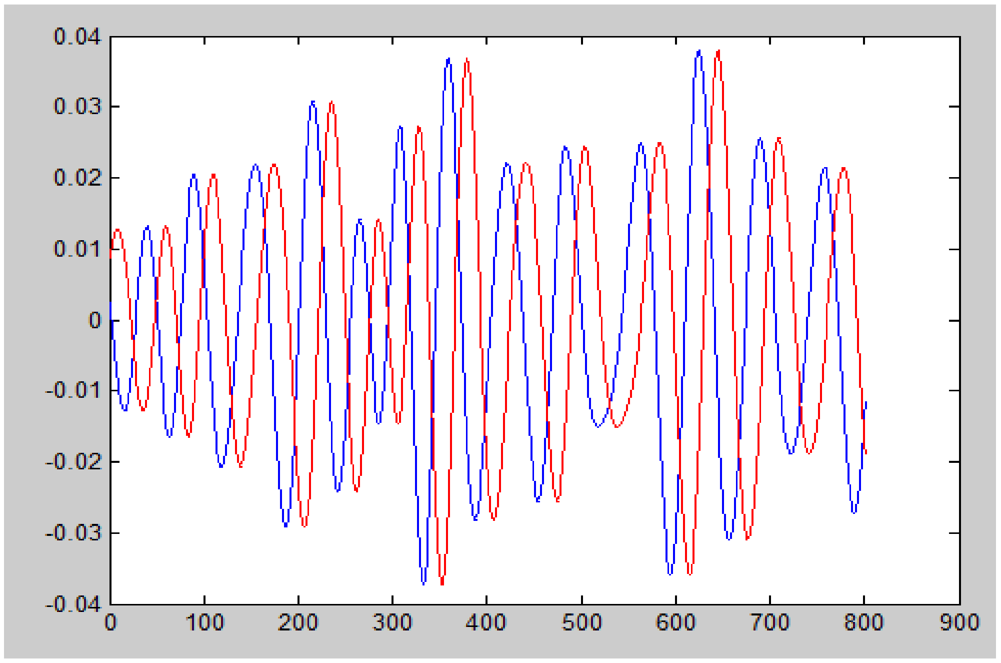


Figure 23: Filtered Signals

The figure 23 shows how the noise is greatly reduced once filtration is applied

4.1.3. TDOA Calculation:

Once the signals are filtered, the TDOA is calculated, the result of the GCC algorithm is illustrated in the figure,

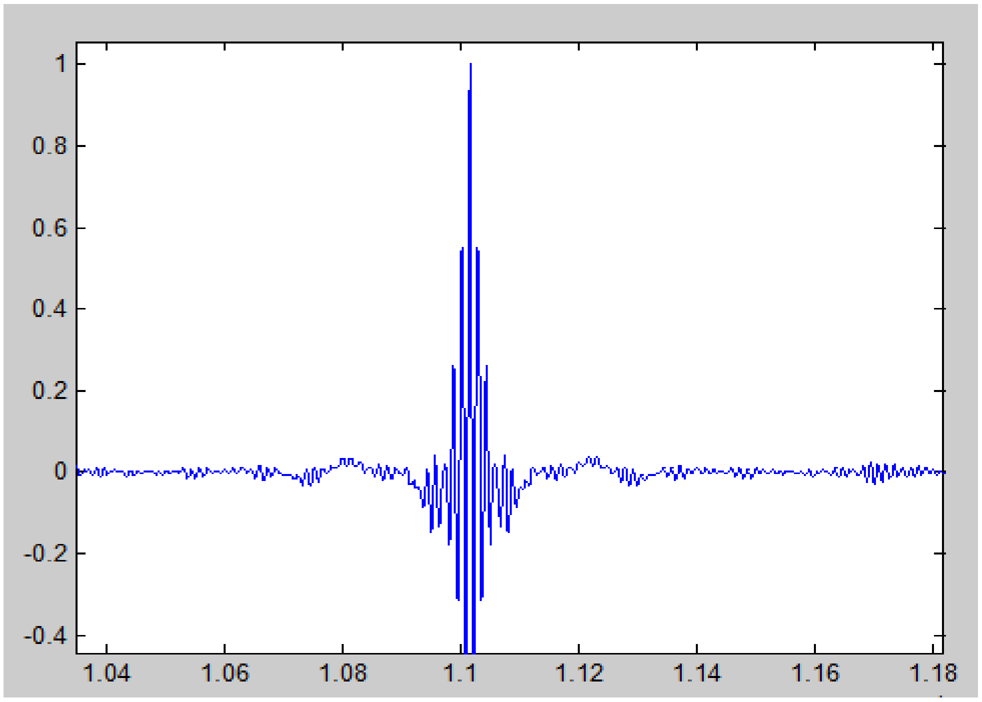


Figure 24: GCC plot

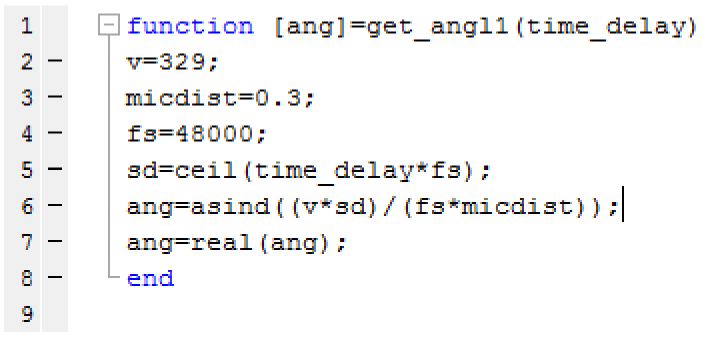
In the figure 24 , the peak is located at the point of the maximum correlation, this peak's index is actually the sample delay, this sample delay is then converted to time delay and then to AOA.

4.1.4. AOA:

The angle of arrival is simply calculated by the relation,

$$\boxed{\cos\varphi = \sin\theta = \frac{c\Delta T_{ij}}{\|\vec{X}_{ij}\|}}$$

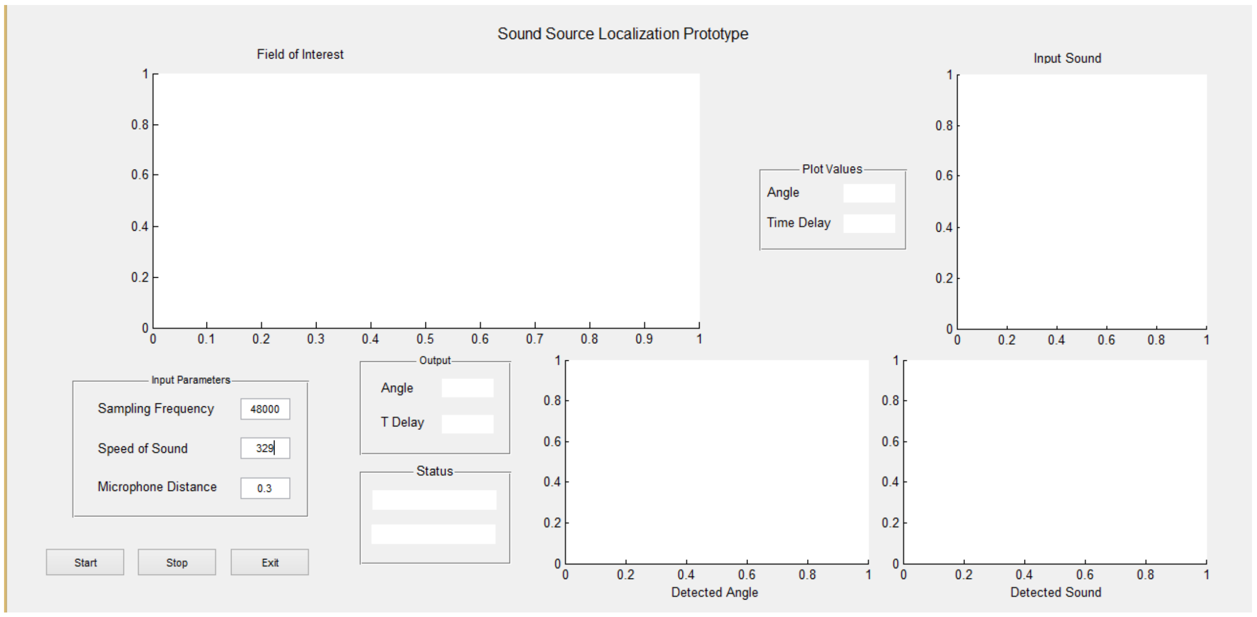
The microphone array and the source of sound are considered to make a right-angled triangle. The AOA is then simply the 'phi' or the 'theta'. The code is implemented on MATlab is follows,



```
function [ang]=get_angl1(time_delay)
v=329;
micdist=0.3;
fs=48000;
sd=ceil(time_delay*fs);
ang=asind((v*sd)/(fs*micdist));
ang=real(ang);
end
```

4.1.5. MATlab GUI:

The AOA and the real time video stream are both shown on the GUI, designed on MATlab using the GUIDE tool.



Sampling Frequency 48000
Speed of Sound 329
Microphone Distance 0.3

Figure 25: MATlab GUI

The figure 25 shows the MATlab GUI, it has the following parts,

1. Field of interest : Displaying the real time video of area of interest
2. Detected angle : This field show the angle in form of a compass
3. Input sound : Displays the real time sound, being read by the microphones
4. Detected sound : Shows the actual frame of sound being processed
5. Status : Shows either ‘detecting’ or ‘listening’

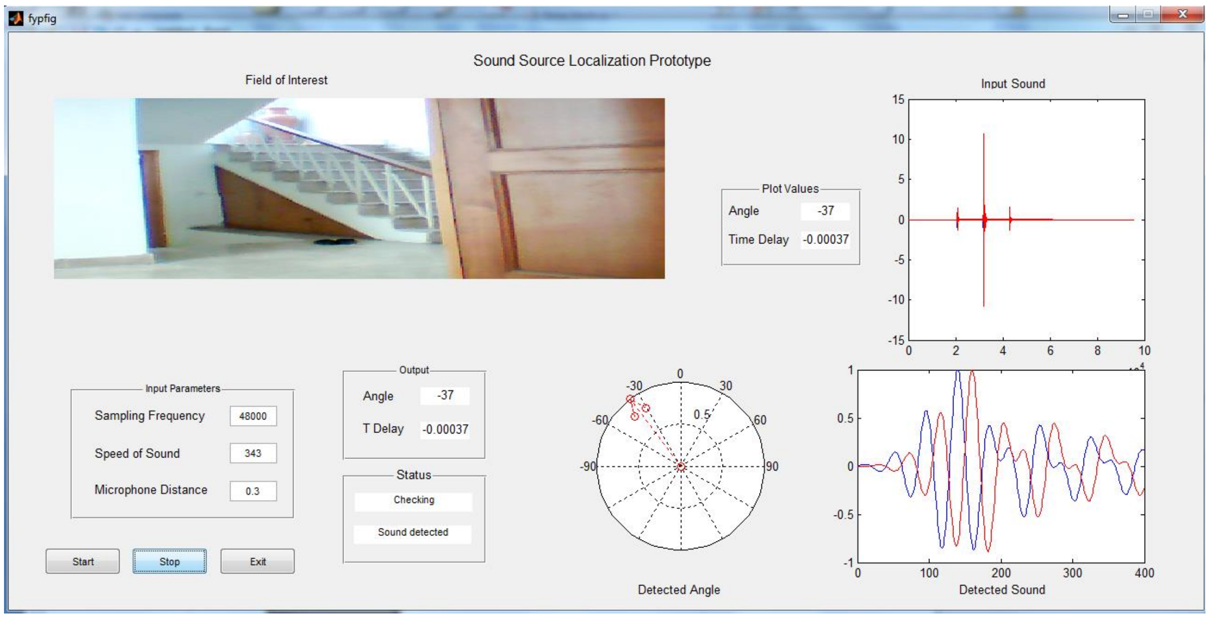


Figure 26: Actual working GUI

This figure 26 shows the working GUI.

Chapter 5

5.1. Recommendations and future work:

1. The system prototype has been implemented on MATlab running on Laptop, it may be progressively designed on to a specific hardware like an FPGA kit or a DSP kit, to make it more efficient and modular.'
2. The microphone used are the basic sound sensors, these may be replaced with the more expensive pressure sensors, to increase range and sensitivity.
3. A PTZ camera may be used to follow the target if it is moving.
4. Multiple systems like this prototype may be employed to get the exact coordinates of the target
5. Increasing the number of arrays in each plain may enable elevation angle calculation as well
6. Increasing the number of microphones per array may yield greater resolution of the calculated angle
7. Robust signal processing , if employed, may result in detecting a singular source amongst many
8. Multiple sound sources may also be located with a single system

5.2. Conclusion:

1. The sound source localization system ensures the timely and accurate response of the QRF and law enforcement agencies
2. Helping in pinpointing the target in real time enables the action to be swift and rapid
3. In a conference room, voices may be localized and the actual microphone and camera may point towards the speaker.
4. In the battle field scenario, the prototype would enable the localization of snipers
5. Bomb blasts can be readily located
6. It is a passive technique, therefore is not detectible

5.2.1. Drawbacks

1. The system is able to successfully localize impulsive and intense sounds, the testing is carried out by clapping firmly, and the system locates the clap noises successfully.
2. The environment plays a crucial role in detection, very noisy and reverberant environments play havoc with the calculations, this has been realized through extensive testing, and this may be remedied by employing more expensive equipment.
3. The system is designed on a laptop, replacing the laptop with a dedicated embedded system would enable more efficient operation and increase portability

5.2.2. Advantages

1. The system is calibrated for detection of impulsive sounds, at present ‘clapping’ noise.
2. The system is able to localize voice, gunshots etc.
3. The system visually displays the area from where the noise is heard.
4. A passive technique, invisible to the enemy
5. Locate target using its acoustic signature, cannot be evaded like radar

References

[1] 'Maximum Likelihood Sound Source Localization and Beamforming for Directional Microphone Arrays', Cha Zhang, Dinei Florêncio, Demba E. Ba, and Zhengyou Zhang, Multimedia, IEEE Transactions on (Volume:10 , Issue: 3), April 2008

[2] ' Sound Source Localization: Microphone Array Design and Evolutionary Estimation', N. M. Kwok, J. Buchholz , , 2005. ICIT 2005. IEEE International Conference on Industrial Technology 14-17 Dec. 2005, Hong Kong

[3] 'The Linear Method for Acoustical Source Localization (Constant Speed Localization Method)' – A Discussion of Receptor Geometries and Time Delay Accuracy for Robust Localization, Sergio R. Buenafuente and Carmelo M. Militello, University of La Laguna (ULL,) Spain

[4] 'Acoustic Source Localization Using Time Delay Estimation', Supercomputer Education and Research Center, Indian Institute of Science, Ashok Kumar Tellakula, August 2007

[5]'Performance optimization of GCC-PHAT for delay and polarity correction under real world conditions', Centre for Digital Music, Queen Mary, University of London, London,

[6] 'Subsample Time Delay Estimation via Improved GCC PHAT Algorithm' Bo Qin, Heng Zhang, Qiang Fu, Yonghong Yan

[7] S. L. Marple, Estimating group delay and phase delay via discrete-time analytic cross-correlation, IEEE Trans. Signal Processing, Vol. 47 (1999) 2604-2607

[8] X. Lai, H.Torp, Interpolation methods for time-delay estimation using cross-correlation method for blood velocity measurement, IEEE Trans. Ultrason, Ferroelect., Freq.Contr., Vol. 46 (1999), 277-290

[9] H. –H. Chiang, C.L.Nikias, A new method for adaptive time delay estimation for non-Gaussian signals, IEEE Trans. Acoust. Speech Signal Process. 38 (1990) 209-219

[10] L. Zhang, Xiaolin Wu, On the application of cross correlation function to subsample discrete time delay estimation, Digital Signal processing. 16 (2006) 682-694

[11] T.Laakso, V.Valimaki, M.Karjalailen and UK Lainie,splitting the unit delay – tools for fractional delay filter design, IEEE Signal Processing Magazine, 1996

Appendix A

Sound source localization

Extended Title:

Real time sound source localization system

Brief Description of The Project / Thesis with Salient Specs:

The project aims at locating an acoustic source by means of microphones, for obtaining input (sound), and methods such as time delay estimation(time delay on arrival, TDOA) and accumulated correlation for establishing the exact bearings of the source.
The acoustic source produces sound waves which are received by multiple microphones placed in such a way that they have some distance between them. The distance between the microphones creates phase delay in the received sound waves, although originating from a single source. This phase delay enables us to triangulate the exact location of the acoustic source.
The input obtained from the microphones would be integrated with a Laptop using interfacing circuitry. The processing would take place on MATlab in the laptop, using user defined methods to pinpoint the target.

Scope of Work

The project will render the detection, of acoustic sources, easier. The sensor would locate the target In real time, by providing real time processing of the multiple sounds In the input and separating the sounds related to our target. Thus enabling the detection of that particular source.

Academic Objectives :

The project will impart following skills into the syndicate:
1. Signal processing techniques
2. MATlab language learning
3. Digital signal processing (sound)
4. Localizing acoustic sources.

Application / End Goal Objectives :

The aim of this system is to provide an accurate mode for locating a target, using sound waves. This is a passive technique and, unlike radar systems, cannot be detected. This system has vast military application for detecting and localizing enemy combatants and/or locating artillery batteries. Civilian applications include integration with existing security system to detect acts of theft or terrorism by locating the source of gunfire or movement and using surveillance to visually highlight the intruders.
Our objective is to demonstrate the acoustic localization system and its application by integrating a video camera to visually highlight the source's locale.

Previous Work Done on The Subject :

- Real-time localization of sound source using an array of acoustic sensors. -2013(EME)
- Wireless sound ranging system -2001(MCS)
- S. T. Birchfield**, A Unifying Framework for Acoustic Localization**, Proceedings of the 12th European Signal Processing Conference (EUSIPCO), Vienna, Austria, September 2004
- S. T. Birchfield and D. K. Gillmor, **Fast Bayesian Acoustic Localization**, Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Orlando, Florida, May 2002

Material Resources Required:

- Laptop
- Microphones
- LCD monitor
- Tripod stand
- Camera
- Wires
- Misc

No of Students Required : 3

Special Skills Required :

- Digital signal processing
- Circuit designing and soldering
- MATlab coding

Appendix B

MATlab code used in the project

1. GUI

```
function varargout = fypfig(varargin)
% FYPFIG MATLAB code for fypfig.fig
%      FYPFIG, by itself, creates a new FYPFIG or raises the existing
%      singleton*.
%
%      H = FYPFIG returns the handle to a new FYPFIG or the handle to
%      the existing singleton*.
%
%      FYPFIG('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FYPFIG.M with the given input arguments.
%
%      FYPFIG('Property','Value',...) creates a new FYPFIG or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before fypfig_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to fypfig_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help fypfig

% Last Modified by GUIDE v2.5 20-May-2015 10:53:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @fypfig_OpeningFcn, ...
                   'gui_OutputFcn',  @fypfig_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before fypfig is made visible.
function fypfig_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to fypfig (see VARARGIN)

% Choose default command line output for fypfig
handles.output = hObject;

%initializing cams
global vid vid2 vid1;
vid=videoinput('winvideo',1,'YUY2_640x480');
vid2=videoinput('winvideo',3,'YUY2_640x480');
vid1=videoinput('winvideo',2,'YUY2_640x480');

%cam1

%axes(handles.axes4);
%vid=videoinput('winvideo',1,'YUY2_160x120');
%himage=image(zeros(160,120,3),'Parent',handles.axes4);
%preview(vid,himage);

%cam2

%axes(handles.axes5);
%vid1=videoinput('winvideo',2,'YUY2_160x120');
%himage1=image(zeros(160,120,3),'Parent',handles.axes5);
%preview(vid1,himage1);

%cam3

%axes(handles.axes6);
%vid2=videoinput('winvideo',1,'YUY2_160x120');
%himage2=image(zeros(160,120,3),'Parent',handles.axes6);
%preview(vid2,himage2);



% Update handles structure
guidata(hObject, handles);

% UIWAIT makes fypfig wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = fypfig_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in startButton.
function startButton_Callback(hObject, eventdata, handles)
% hObject    handle to startButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global exitter;
exitter=1;
global vid vid2 vid1;
Fs=str2num(get(handles.Fsbox,'String'));
vs=str2num(get(handles.vsbox,'String'));
dist=str2num(get(handles.MicBox,'String'));
chk=0;
%breaker=0;
while (exitter==1)%for i=1:1:3
    %set(handles.breaker,'String',num2str(breaker));
    %if (breaker<3)
        chk=0;
        set(handles.status,'String','Listening');
        pause(0.5)
        [left,right]=starrt1(Fs);
        [left_act,right_act,chk]=detec_t2(left,right);
        set(handles.status,'String','Checking');
        axes(handles.axes3)
        plot(left)
        hold on
        plot(right,'r')
        hold off
        if(chk==0)
            time_delay=99;
            anglee=99;
            %breaker=breaker+1;
            set(handles.angletextdisp,'String','Nil');
            set(handles.tdelaytextdisp,'String','Nil');
            set(handles.doing,'String','No sound');
            axes(handles.axes1)
            plot(xlim,[-0.5 -0.5])
            hold on
            plot(xlim,[0.5 0.5],'r')
            hold off
            axis([0 1 -1 1])

        else
            time_delay=get_time1(left_act,right_act,Fs);
            %anglee=get_angl(time_delay,vs,dist,Fs);
            anglee=look_up(time_delay);
            %breaker=0;
            a=field_chk(anglee);

            axes(handles.axes5);
            switch (a)
                case 1
```

```
                    himage=image(zeros(640,480,3),'Parent',handles.axes5);
                    preview(vid2,himage);
                case 2

                    himage1=image(zeros(640,480,3),'Parent',handles.axes5);
                    preview(vid,himage1);
                case 3

                    himage2=image(zeros(640,480,3),'Parent',handles.axes5);
                    preview(vid1,himage2);
            end
            set(handles.previousangle,'String',num2str(anglee));
            set(handles.previoustime,'String',num2str(time_delay))
            set(handles.angletextdisp,'String',num2str(anglee));
            set(handles.tdelaytextdisp,'String',num2str(time_delay));
            set(handles.doing,'String','Sound detected');
            axes(handles.axes1)
            plot(left_act)
            hold on
            plot(right_act,'r')
            hold off
            axes(handles.axes2)
            [x,y]=adj_angle(anglee);
            compass(x,y,':or')
            % Altering the angular label
            set(findall(gcf, 'String', '210'),'String', '');
            set(findall(gcf, 'String', '240'),'String', '');
            set(findall(gcf, 'String', '270'),'String', '');
            set(findall(gcf, 'String', '300'),'String', '');
            set(findall(gcf, 'String', '330'),'String', '');
            set(findall(gcf, 'String', '180'),'String', '-90');
            set(findall(gcf, 'String', '150'),'String', '-60');
            set(findall(gcf, 'String', '120'),'String', '-30');
            set(findall(gcf, 'String', '90'),'String', '0.1');
            set(findall(gcf, 'String', '60'),'String', '31');
            set(findall(gcf, 'String', '30'),'String', '60');
            set(findall(gcf, 'String', '0'),'String', '90');
            set(findall(gcf, 'String', '0.1'),'String', '0');
            set(findall(gcf, 'String', '31'),'String', '30');
            % Altering the radial label
            set(findall(gcf, 'String', '  0.4'),'String', ' ');
            set(findall(gcf, 'String', '  0.2'),'String', ' ');
            set(findall(gcf, 'String', '  0.6'),'String', ' ');
            set(findall(gcf, 'String', '  0.8'),'String', ' ');
            set(findall(gcf, 'String', '  1'),'String', ' ');
        end
        %pause(3)
    %else
    %       break
    %end
end

% --- Executes on button press in stopButton.
function stopButton_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to stopButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global exitter;
exitter=0;




function Fsbox_Callback(hObject, eventdata, handles)
% hObject    handle to Fsbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Fsbox as text
%        str2double(get(hObject,'String')) returns contents of Fsbox as a double



% --- Executes during object creation, after setting all properties.
function Fsbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Fsbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function vsbox_Callback(hObject, eventdata, handles)
% hObject    handle to vsbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vsbox as text
%        str2double(get(hObject,'String')) returns contents of vsbox as a double



% --- Executes during object creation, after setting all properties.
function vsbox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vsbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double



% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function MicBox_Callback(hObject, eventdata, handles)
% hObject    handle to MicBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MicBox as text
%        str2double(get(hObject,'String')) returns contents of MicBox as a double



% --- Executes during object creation, after setting all properties.
function MicBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MicBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when uipanel4 is resized.
function uipanel4_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in exitButton.
function exitButton_Callback(hObject, eventdata, handles)
% hObject    handle to exitButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%stoppreview(obj1);
%stoppreview(obj2);
%stopstoppreview(obj3);
close all;
```

2. Recording function

```
function [left,right]=starrt1(fs)



%%% recording

dev=audiodevinfo;
recorddd=audiorecorder(fs,8,2,dev.input(5).ID);
recordblocking(recorddd,1.5);
xx=getaudiodata(recorddd);
x=xx(:,1);
y=xx(:,2);

%%% filteration

filt=shaniaaaf1;
xf=filter(filt,x);
yf=filter(filt,y);



left=xf.*1000;
right=yf.*1000;

end
```

3. TDOA calculation

```
function time_delay=get_time1(left,right,fs)
m=length(left);
fftlength=2*m;
n=(-m/2)+1:m/2;
[Sxx, Fxx] = pwelch(left,[],[],fftlength,fs);
[Syy, Fyy] = pwelch(right,[],[],fftlength,fs);
[Sxy, Fxy] = cpsd(left,right,[],[],fftlength,fs);
[corr1, t1] = GCC('unfiltered',Sxx, Syy, Sxy,fs,length(n));
[~,k] = max(corr1);
k = t1(k);
time_delay=k;
end
```

4. GCC function

```
function [G,t,R] = GCC(m,Pxx,Pyy,Pxy,Fs,frame,N)
% check input arguments
if nargout==1 & nargin<4
error('Wrong number of input arguments');
elseif nargout==2 & nargin<6
error('Wrong number of input arguments to compute time ticks');
elseif nargin<7
N = size(Pxy,1);
end
% number of channels
Nchn = size(Pxy,2);
% define pre-whitening filter
% time cross-correlation.
W = ones(N,Nchn);

% apply the filter
R = Pxy .* W;
% estimate the generalized cross-correlation (GCC)
G = fftshift(real(ifft(R)),1);
% NB: the real part is extracted to avoid the small undesidered imag.
for k=1:Nchn
G(:,k) = G(:,k)/max(abs(G(:,k)));
end
if nargout>1
% calculate thick along time axis
resolution = (Fs*N)/(2*frame);
if mod(N,2)==0
t = [-N/2 : (N/2)-1] / resolution;
else
t = [-(N-1)/2 : (N-1)/2] / resolution;
end
end
```

5. Angle calculation

```
function [ang]=get_angl1(time_delay)
v=329;
micdist=0.3;
fs=48000;
sd=ceil(time_delay*fs);
ang=asind((v*sd)/(fs*micdist));
%s=v*time_delay;
%ang=asind(s/micdist);
%ang=ang*(180/pi)
%ang=acosd(s/micdist);
%ang=abs(ang);
ang=real(ang);
end
```

Appendix C

Time Line

Literature Study	Completed						
Array Design			Completed				
MATlab Code					Completed		
Interfacing					Completed		
Camera integration						Completed	
MATlab GUI						Completed	
Finalization							Completed
	November	**December**	**January**	**February**	**March**	**April**	**May**

Appendix D

Cost Breakdown

Component	Quantity	Unit price	Price
Microphones	6	400	2400
Aux cables	7	120	840
USB HUB	2	500	1000
Tripod stand	1	3500	3500
USB cameras	3	1200	3600
Miscellaneous	wires etc.	2500	2500
Total		-	13840/-