

# PAINTING WITH MUSIC

by

Weijian Zhou

A thesis submitted in conformity with the requirements  
for the degree of Master of Applied Science and Engineering  
Graduate Department of Electrical and Computer Engineering  
University of Toronto

© Copyright 2015 by Weijian Zhou

# **Abstract**

Painting with Music

Weijian Zhou

Master of Applied Science and Engineering

Graduate Department of Electrical and Computer Engineering

University of Toronto

2015

In this paper, we described a small area sound localization system with two microphone arrays. We evaluated different array architectures and demonstrated that a two array system outperforms a single array system of similar physical dimension. The proposed system localizes in 80 milliseconds and achieved an average error of less than 3 cm for both point localization and movement tracking in a local one meter by one meter region.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>4</b>
2.1	Point localization . . . . .	4
2.1.1	ILD . . . . .	4
2.1.2	LTM . . . . .	6
2.1.3	TDOA . . . . .	6
2.2	Movement Tracking . . . . .	11
2.2.1	Averaging filtering . . . . .	11
2.2.2	Kalman filtering . . . . .	11
<b>3</b>	<b>Array Architecture</b>	<b>12</b>
3.1	Two Microphones . . . . .	12
3.2	Three Microphone Array . . . . .	15
3.2.1	Linear Configuration . . . . .	16
3.2.2	Equilateral Triangular Configuration . . . . .	17
3.3	More than three . . . . .	18
3.4	Discussion . . . . .	19
<b>4</b>	<b>Experiment</b>	<b>20</b>
4.1	System . . . . .	20

4.2	Setup . . . . .	24
4.2.1	Point localization . . . . .	24
4.2.2	Movement tracking . . . . .	25
4.3	Results . . . . .	28
4.3.1	Point localization . . . . .	28
4.3.2	Movement tracking . . . . .	32
4.3.3	Discussion . . . . .	38
<b>5</b>	<b>Conclusion</b>	<b>40</b>
	<b>Bibliography</b>	<b>42</b>

# List of Figures

3.1	Uncertainty region. Yellow dots represent microphones' locations and the white dot represents the location of the source. . . . .	13
3.2	Uncertainty region. Microphones are at the vertices of a 20cm equilateral triangle. The source is 20cm away from the array. . . . .	14
3.3	Uncertainty region. Microphones are at the vertices of a 20cm equilateral triangle. The source is 80cm away from the array. . . . .	15
3.4	3 microphones are placed in a line, 10 cm apart from each other. The average error is 55.05 cm . . . . .	16
3.6	Error heatmap for different array configurations. The heatmap scale is the intersection area measured in $cm^2$ . . . . .	18
4.1	Localization system setup . . . . .	21
4.2	Likelihood maps for localization. The source is placed at $(0.0, -0.3)$ m .	22
4.3	Setup for point localization evaluation . . . . .	25
4.4	Setup for movement tracking evaluation . . . . .	26
4.5	Localization error versus window size . . . . .	28
4.6	Computation time versus window size . . . . .	29
4.7	error distribution in the grid. Arrays are placed at $(-0.5 \text{ m}, 0 \text{ m})$ and $(0.5 \text{ m}, 0 \text{ m})$ . . . . .	30
4.8	Localization error as the distance between the source and the microphone arrays increases. The source is placed on the $y$ axis. . . . .	31

4.9	Localization quality versus window size . . . . .	32
4.10	Localization error versus window size . . . . .	33
4.11	white noise (10 cm per second) . . . . .	34
4.12	music A (10 cm per second) . . . . .	35
4.13	music B (10 cm per second) . . . . .	36
4.14	white noise (20 cm per second) . . . . .	37
4.15	music A (20 cm per second) . . . . .	38
4.16	music B (20 cm per second) . . . . .	39

# Chapter 1

## Introduction

Accurate indoor localization allows creation of novel applications with surrounding awareness that uses position and movement information as input.

Global Positioning System (GPS) is the prevailing technology used for outdoor localization. A set of 32 satellites equipped with synchronized atomic clock orbit around the earth and broadcast their time and position information at fixed intervals. A GPS receiver listens to at least 4 satellites and uses the broadcasted timing information to infer the location. Commercial grade GPS has an average error of a few meters, depending on the size and quality of the receiver [1]. While accuracy in this range is good for many applications including driving navigation and vehicle tracking, it does not provide enough precision for local movement tracking.

Ultrasound based indoor localization approaches, on the other hand, have achieved sub-centimeter accuracy [4]. These systems employ multiple ultrasound receivers and use the arrival time difference between the sound source and receivers to infer the source location. Although obtaining good accuracy, ultrasound systems require the use of expensive transducers.

Bluetooth and Wi-Fi based technologies have gained popularity in indoor positioning recently, mainly due to the widespread deployment of bluetooth tags and Wi-Fi stations in

public spaces. The fact that modern consumer devices such as mobile phones and tablets are commonly equipped with Wi-Fi and bluetooth modules also made this approach particular attractive because no hardware needs to be installed on the user end. Recent applications including serving advertisements and coupons to consumers in shopping centers based on the customer's location. In these systems, signal strength received from different base stations are used for the estimation of the device location. However, their reported accuracy are in the range of 1 to 5 meters [2, 3], which is not enough for local movement tracking.

In this work, we have built a source localization system that is portable, inexpensive, yet reasonably accurate for localization in a small area. To test the accuracy of our localization system, we have acoustic localization in an one meter by one meter area. The experiment area size is adequate for our purposes because human arm movement are constrained in this area. Our experiment show that our system reports reasonably good accuracy in this area. Our system is built with inexpensive electret microphones mounted on portable frames. Users can interact with our system using any device that has audio output such as a mobile phone.

This system can be used in Human-Computer Interaction (HCI) applications that incorporate sound position and movement information. One can use this system to build virtual drawing applications where the user can draw with music without physically touching the computer. One can also use this system to design interactive AI games with physical pieces. For example, a chess game with physical pieces can be developed where each piece is equipped with a motor and a music tag. Since the computer knows where all the pieces are and which one the user has moved, it can make its corresponding move. Another similar example is to build an AI toy car racing game, where the player controls one car and the computer controls another. With real time location information, the AI engine can compete with the player on a racing track.

This system can also be used to in Augmented Reality (AR) applications. For ex-

ample, the user can wear a music tag on one finger, and then the system would be able to track the finger movement. This particular setup can be used as a virtual mouse for wearable technologies such as Google Glass.

In this paper, we first discussed in Chapter II some prior relevant research and approaches in sound localization. In Chapter III, we evaluated different array architectures and their impact on localization accuracy. We demonstrated that a two array system outperforms a single array system of similar physical dimension. In Chapter IV, we first presented the chosen architecture along with hardware details, followed by experiment details and results were presented.

# Chapter 2

## Background

### 2.1 Point localization

Acoustic localization has been researched extensively in the literature. Localization techniques can be broadly categorized into Interaural Level Difference (ILD), Location Template Matching (LTM), and Time Difference of Arrival (TDOA) based approaches.

#### 2.1.1 ILD

ILD techniques rely on the observation that signal intensity decays as the distance to the microphone increases. A microphone closer to the signal source would receive the signal with higher intensity than a microphone farther away. With multiple microphones, it is possible to infer the source location by comparing the signal intensity received at different microphones. Human's auditory system has used ILD cues to infer source direction [6], and this technique is most effective to localize high frequency sources, because they don't diffract effectively around the listener's head and produce a significant intensity difference.

For a point sound source in a direct field, the signal intensity decays proportional to the square of the distance between the source and the microphone. Let  $I_i$  denote the

received signal intensity at microphone  $i$ :

$$I_i \propto \frac{1}{d_i^2} I_s \quad (2.1)$$

where  $d_i$  is the distance between the audio source and microphone  $i$ , and  $I_s$  is the sound intensity at the source. With two microphones, the signal intensity received at both microphones has to satisfy equation 2.1:

$$I_1 d_1^2 = I_2 d_2^2 \quad (2.2)$$

$$\frac{I_1}{I_2} = \frac{d_2^2}{d_1^2} \quad (2.3)$$

It can be shown that, on a 2D plane, points satisfy equation 2.3 form a circle when  $I_1 \neq I_2$  and form a line when  $I_1 = I_2$  [5]. With two microphones all points on this curve generate the same intensity ratio and can not be distinguished from each other. Multiple approaches have been investigated to eliminate this ambiguity. [5] employed multiple microphones and used the intersection of circles from each microphone pair to estimate the source location. [7] combined ILD with Interaural Time Difference (ITD) to estimate the source direction (i.e azimuth). Instead of solving the intersection of circles, [8] employed machine learning techniques to automatically learn the mapping from ILD and ITD features to location coordinate. It uses four microphones and requires a training phase, during which the sound source is manually placed at predetermined locations for the system to learn the parameters that map from feature space to the sound location.

ILD approach relies on the accurate measurement of the received intensity ratio between microphone pairs. Any obstacle object between the sound source and any microphone would produce a significant distortion in the measured intensity ratio. We find this approach to be too restrictive for our system, since in an interactive system we can not control whether or not the user places any obstacle between the sound source and the microphones.

### 2.1.2 LTM

In LTM based approaches, acoustic templates acquired from different locations are first stored in the system during a “training” phase. Localization can be performed by comparing the incoming waveform with the stored templates, and the location with the best matching template is chosen as the output. Authors in [9] has built a virtual piano keyboard application where different musical notes are associated with different locations on a surface. A user can then play by tapping at different locations on the surface. Different ways of extracting templates from raw acoustic source and different similarity measures have been investigated in the past.

[9] and [10] investigated using max value from cross-correlation as a similarity measure to localize user taps on interactive surfaces. [11] used L2 distance in the Linear Predictive Coding coefficient space as a similarity measure to localize taps on surfaces. [12] further explored accuracy improvement by using multiple templates for each location and speed improvement by merging multiple templates into one representative template.

The requirement of having a template for each location to be detected makes this approach too restrictive for our project, since we want the localization to be continuous in a 2D region. [12] has investigated contiguous localization by linearly extrapolating between stored locations, but result has high error. Moreover, the need to recalibrate all locations during setup is too cumbersome for the end users in a portable system. Therefore, our main focus will be on TDOA based approaches.

### 2.1.3 TDOA

TDOA approaches exploit the difference of arrival time between the acoustic source and two fixed microphones on the plane. It can be easily shown that the acoustic sources with the same TDOA to two fixed microphones on the plane form a hyperbola. When you have more than two microphones, each pair would give a different hyperbola. The intersection of all the hyperbolas marks the source location. TDOA approaches rely on

accurate estimates of arrival time differences between microphones.

In [13], authors used eight microphones mounted on the corners of a ping pong table to localize points where the ball hits the table. They used a threshold to determine the arrival time of acoustic signal. This approach works well in noise free environment but the performance degrades with background noise. Their approach also suffers from dispersive deflections that arrive before the main wavefront of the acoustic signal. To make it more robust, authors in [19] and [20] extracted descriptive parameters for each significant peak(e.g., peak height, width, mean arrival time). The algorithm then used extracted parameters to predict arrival time with a second order polynomial, the parameters of which were fitted during calibration at fixed locations. Authors in [14] have used similar techniques and built an interactive window by placing four microphones on four corners of a glass pane. The glass window was installed in shopping centers. A projector projects product information onto the glass where consumers can browse between pages by tapping on the window.

Cross-correlation has also been used to measure signal arrival time differences[15, 16, 18]. Cross correlation is a measure of similarity between two signals. For real valued signals  $x_1(t)$  and  $x_2(t)$ , the cross-correlation between them at a particular time shift  $\tau$  can be calculated as:

$$R_{x_1,x_2}(\tau) = \int_{-\infty}^{\infty} x_1(t)x_2(t + \tau)dt \quad (2.4)$$

We can take Fourier Transform on both sides of equation 2.4:

$$\mathcal{F}\{R_{x_1,x_2}(\tau)\} = X_1(\omega)X_2(-\omega) \quad (2.5)$$

$$= X_1(\omega)X_2(\omega)^* \quad (2.6)$$

Where  $X_1(\omega)$  and  $X_2(\omega)$  are the Fourier Transforms of  $x_1(t)$  and  $x_2(t)$ . We can retrieve

the cross-correlation result in time domain by taking inverse Fourier transform:

$$R_{x_1, x_2}(\tau) = \int_{-\infty}^{\infty} X_1(\omega) X_2(\omega)^* e^{j\omega\tau} d\omega \quad (2.7)$$

The arrival time difference  $t_0$  is the time shift  $\tau$  that maximizes 2.7:

$$t_0 = \arg \max_{\tau} R_{x_1, x_2}(\tau) \quad (2.8)$$

The benefits of calculating cross-correlation in the frequency domain as shown in equation 2.7 are two folds. The first benefit is to speedup the calculation. Calculating cross-correlation using equation 2.4 requires multiplying and summing the two signal vectors for each time shift  $\tau$ . With discrete signals of length  $n$ , it will take  $O(n^2)$  number of calculations. Doing the same calculation in frequency domain, we need to transform the signal into frequency domain, multiple and sum the two transformed signal vectors once and then transform the result back to the time domain. Transforming a signal from time domain to frequency domain and back can be done efficiently with Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT), and the amount of calculation needed is  $O(n \log n)$ . Multiply and sum the transformed signal vectors takes another  $O(n)$ . Therefore, the total calculation required to calculate cross-correlation using Fourier transform is  $O(n \log n)$ , which is asymptotically faster than calculating in the time domain. This calculation speedup is especially beneficial in real-time interactive systems since significant time lag would produce unsMOOTH user experience.

The second benefit of formulating cross correlation in frequency domain is that it provides a unified framework to prefilter the signals. Cross-correlation with prefiltering is known as *generalized cross correlation (GCC)*. Different prefiltering approaches have been investigated to improve arrival time difference estimation [21, 22, 23]. Under the GCC framework, the arrival time difference  $t_0$  between two signals  $x_1(t)$  and  $x_2(t)$  is

estimated as:

$$t_0 = \arg \max_{\tau} R_{x_1 x_2}(\tau) \quad (2.9)$$

$$R_{x_1 x_2}(\tau) = \int_{-\infty}^{\infty} W(\omega) X_1(\omega) X_2^*(\omega) e^{j\omega\tau} d\omega \quad (2.10)$$

$W(\omega)$  provides a way to prefilter signals passed to the cross correlation estimator. We focused on three ways of prefiltering the signal:

**GCC**  $W(\omega) = 1$ . No prefiltering is done. This is unfiltered normal cross correlation.

**GCC\_PHAT**  $W(\omega) = \frac{1}{|X_1(\omega)X_2^*(\omega)|}$ . Each frequency is divided by its magnitude. Only phase information contributes to delay estimation.

**GCC\_PHAT\_SQRT**  $W(\omega) = \frac{1}{|X_1(\omega)X_2^*(\omega)|^{0.5}}$ . This is somewhere between GCC and GCC\_PHAT. Part of magnitude information is included in arrival time difference estimation.

To see the reasoning behind different prefiltering approaches, we separate the magnitude part from the phase part of  $X_1(\omega)$  and  $X_2(\omega)$  in Equation 2.10:

$$R_{x_1 x_2}(\tau) = \int_{-\infty}^{\infty} W(\omega) |X_1(\omega)| |X_2(\omega)| e^{j(\omega\tau - (\angle X_2(\omega) - \angle X_1(\omega)))} d\omega \quad (2.11)$$

$$= \underbrace{\int_{-\infty}^{\infty} W(\omega) |X_1(\omega)| |X_2(\omega)|}_{\text{weighting}} \cos(\Theta_e) d\omega \quad (2.12)$$

Where  $\Theta_e$  is the phase error:

$$\Theta_e = \omega\tau - (\angle X_2(\omega) - \angle X_1(\omega))$$

We can look at the real part of equation 2.11 only since both  $x_1(t)$  and  $x_2(t)$  are real valued signals. When  $\tau$  is the true arrival time difference between  $x_1(t)$  and  $x_2(t)$ , phase error  $\Theta_e = 0$ , and  $\cos(\Theta_e) = 1$ . When  $\tau$  differs from the true arrival time difference,

$\cos(\Theta_\epsilon) < 1$ . Therefore,  $\cos(\Theta_\epsilon)$  can be seen as a measure of the phase error, and  $W(\omega)|X_1(\omega)||X_2(\omega)|$  describes how the error should be weighted at each frequency. The TDOA estimator essentially sums the weighted phase error at each frequency.

Without any prefiltering (i.e  $W(\omega) = 1$ ), the estimator weighs the phase error at each frequency by the magnitude of the signal at that frequency. In this weighting scheme, phase error at frequencies with higher magnitudes are penalized more compared to frequencies with a lower magnitude. This weighting is appropriate if there is only one source present, since frequencies with higher magnitude have higher Signal to Noise Ratio (SNR). It makes sense to place higher weights at frequencies with higher SNR, since low SNR regions can be dominated by noise.

However, with multiple sources, the source with the highest magnitude will dominate the phase error estimation, but there is no particular reason to assign a higher weight to the source with the highest volume. All sources should contribute equally in the phase error estimation. In GCC\_PHAT,  $W(\omega)$  is set to  $\frac{1}{|X_1(\omega)||X_2(\omega)|}$ . In effect it ignores the signal magnitude and weighs phase errors uniformly across frequencies. Since the phase error at every frequency is weighted equally, this technique will suffer from error accumulation if the source has a lot of low power regions in the frequency domain. This weighting is also beneficial if the source signal is white noise, since white noise should contain all frequency components with equal magnitude.

In GCC\_PHAT\_SQRT,  $W(\omega)$  is set to  $\frac{1}{(|X_1(\omega)||X_2(\omega)|)^{0.5}}$ . Phase error weighting at each frequency still depends on the signal strength at that frequency, but the dependency is much weaker than that in unfiltered GCC. On the other hand, this weighting scheme doesn't go to the other extreme of completely ignoring signal strength information as does in GCC\_PHAT. This approach represents a balance between unfiltered GCC and GCC\_PHAT.

## 2.2 Movement Tracking

a

### 2.2.1 Averaging filtering

a

### 2.2.2 Kalman filtering

a

# Chapter 3

## Array Architecture

In this chapter, we first look at how localization accuracy is affected by TDOA accuracy in a two microphone setup. We then explore different microphone array configurations and select one that gives a good localization accuracy while being portable.

### 3.1 Two Microphones

As was mentioned in the previous chapter, points with the same TDOA to two fixed locations form a hyperbola on a 2D plane. However, in practical systems we can only measure TDOA up to a precision. Therefore we look at all points with difference of distance close to some target value within measurement error  $\epsilon$ . This  $\epsilon$  represents accuracy on the measurement of difference of distances, and in practice it is related to sampling rate and estimation techniques. In this chapter we evaluate the impact of difference of distance estimation on localization accuracy.

To see how precision affects localization accuracy, we simulated two microphones placed at:  $M_1 : (x = -10 \text{ cm}, y = 0 \text{ cm})$  and  $M_2 : (x = 10 \text{ cm}, y = 0 \text{ cm})$ . A test sound source is emitted at point  $P$  which is 50 centimeters away from the origin  $(0, 0)$ . Let  $2a$

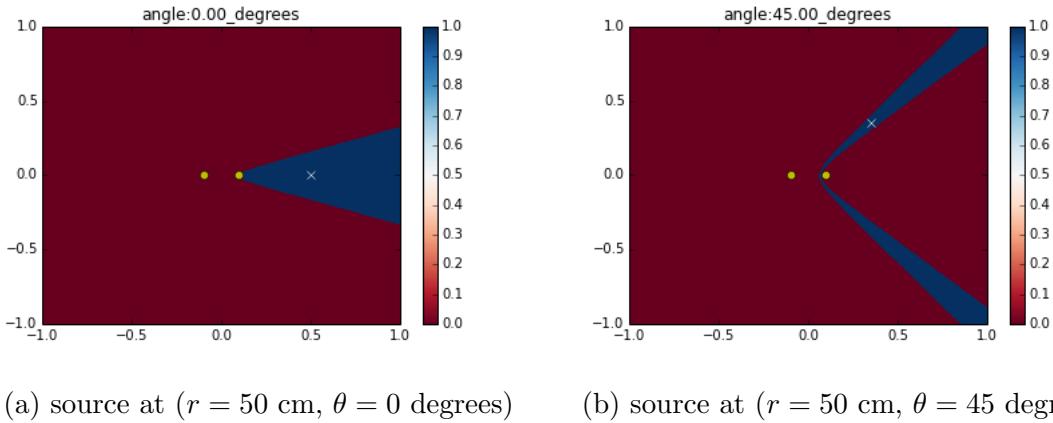


Figure 3.1: Uncertainty region. Yellow dots represent microphones' locations and the white dot represents the location of the source.

denote the TDOA between  $P$  and two microphones:

$$2a = \frac{PM_1 - PM_2}{340 \text{ m/s}}$$

where 340 m/s is used as the speed of sound. Assuming a sampling rate of 34 KHz, fig 3.1 shows the region  $R$  where all points have TDOA close to  $2a$  s within one sample difference:

$$R = \{\hat{P} : \left| \frac{(\hat{P}M_1 - \hat{P}M_2)}{340 \text{ m/s}} - 2a \right| < \frac{1}{2} \text{ samples}\}$$

Intuitively, points in  $R$  have TDOA to two microphones very similar to each other. Looking at fig 3.1, we can still see that  $R$  has the shape of a hyperbola, but with an uncertainty region around it. The thickness of the uncertainty region is not uniform around the hyperbola, the farther away the point is, the larger the uncertainty region becomes. This indicates for the same delta distance movement it will generate smaller TDOA change when the source is farther away from the array. The size of the uncertainty region is also angle dependent: points closer to the line connecting microphones have larger region compared to points close to the line bisecting microphones.

This can also be seen analytically. Assuming two microphones are placed on the x-axis at  $M_1 : (-c, 0)$  and  $M_2 : (c, 0)$ . All points  $P : (x, y)$  with difference of distance

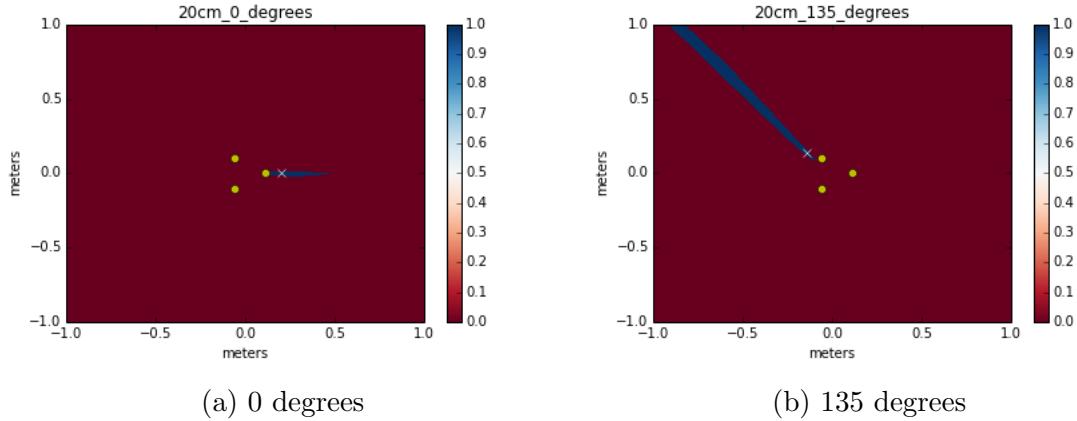


Figure 3.2: Uncertainty region. Microphones are at the vertices of a 20cm equilateral triangle. The source is 20cm away from the array.

$|PM_1 - PM_2| = 2a$  satisfies:

$$\frac{x^2}{a^2} - \frac{y^2}{c^2 - a^2} = 1 \quad (3.1)$$

To see how the difference of distance changes with respect to source location, we can expand the equation and find the partial differential  $\frac{\partial a}{\partial x}$ :

$$\frac{\partial a}{\partial x} = \frac{x(c^2 - a^2)}{a(x^2 + y^2 + c^2) - 2a^3} \quad (3.2)$$

Since all points in equation 3.2 must lie on the hyperbola, we can substitute 3.1 into 3.2:

$$\frac{\partial a}{\partial x} = \frac{c^2 - a^2}{\frac{c^2}{a}x - \frac{a^3}{x}} \quad (3.3)$$

The denominator of equation 3.3 increases monotonically as  $|x|$  increases, which indicates  $\frac{\partial a}{\partial x}$  decreases as we move farther away along the hyperbola. The same distance move  $\delta x$  would generate smaller change in difference of distance  $a$  when the source is farther away from the microphones.

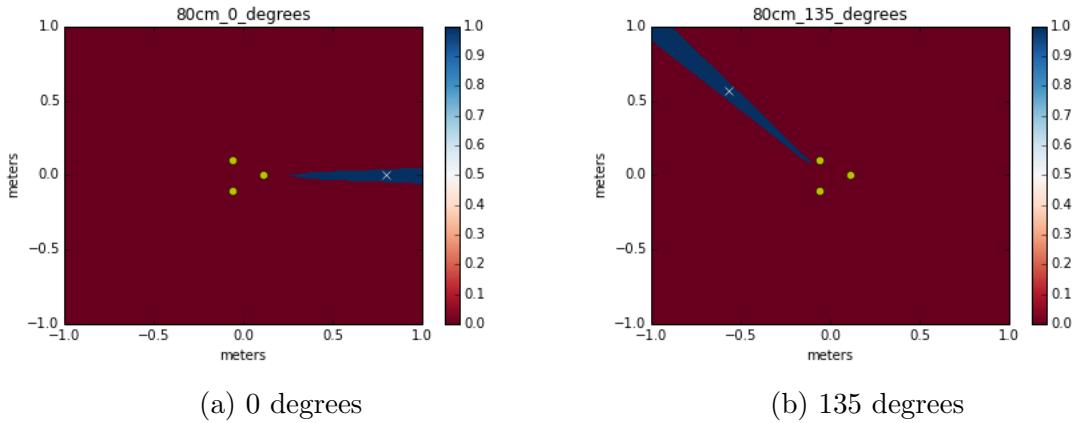


Figure 3.3: Uncertainty region. Microphones are at the vertices of a 20cm equilateral triangle. The source is 80cm away from the array.

### 3.2 Three Microphone Array

With more than two microphones, each pair of microphones generates a hyperbolic region and localization becomes finding the intersection of hyperbolic regions. The smaller the intersection region, the better the localization accuracy. To see how accuracy changes with array placement and sound source location, three microphones are placed at the three vertices of a 20 cm equilateral triangle. An audio source is placed at 20 cm away from the center of the array. Fig 3.2 shows the intersection of regions for 2 different placement of the sound source. It can be seen that accuracy decreases when sound source becomes close to the line connecting any two microphones. This observation is consistent with the two microphone case, since points close to lines connecting microphones have a larger uncertainty region.

To see how sound source distance affects localization accuracy, the same simulation is carried out with the sound source moved from 20 cm to 80 cm away from the center of the array. Results are presented in fig 3.3. Comparing with fig 3.2, accuracy decreases as the distance to the array increases. This is also consistent with our observation in 2 microphone case where sources farther away would result in larger uncertainty region.

Each microphone pair generates a hyperbolic region, and the source location is in

the intersection of these regions. The area of the intersection region is a measure of the localization accuracy: the smaller the area, the more certain we are about the source location. Different array configuration gives different intersection area size. To evaluate an array's accuracy in a region, we can place sound source at predetermined grid points in the region and look at the size of the intersection area for each tested point in the grid. The smaller the intersection area size, the better the configuration. Another measure we can use to evaluate an array configuration is the error distance between the centre of the intersection area to the actual source location. The smaller the error difference on average, the better the configuration. We used both the intersection area size and error distance as measures for evaluating the following configurations.

### 3.2.1 Linear Configuration

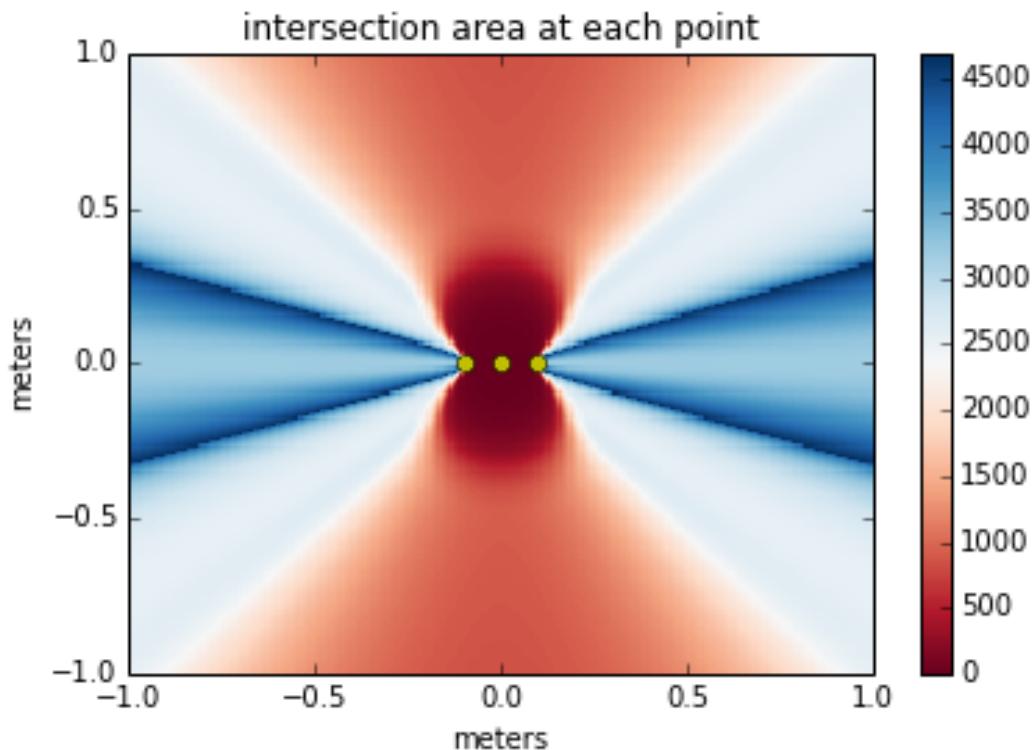
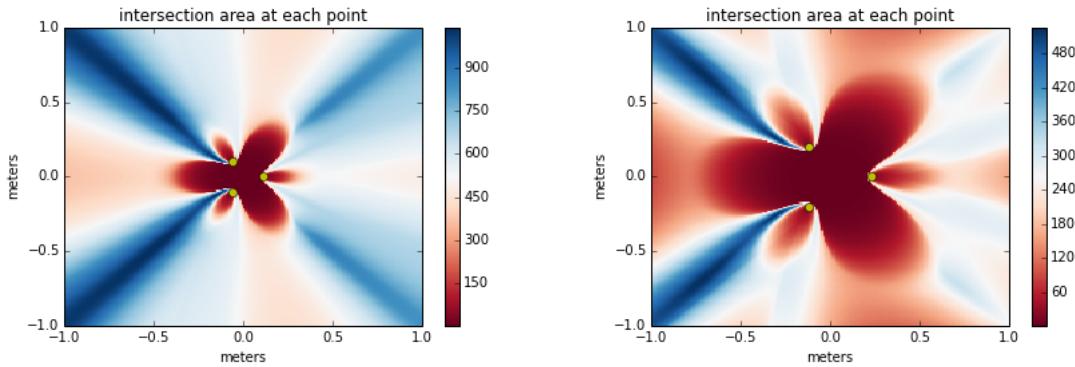


Figure 3.4: 3 microphones are placed in a line, 10 cm apart from each other. The average error is 55.05 cm

In fig 3.4, we experimented with an extreme arrangement where all three microphones are placed on a line, 10 cm apart from each other. Looking at the figure above, we see that the intersection area is greater than  $3000\text{cm}^2$  for points along the x axis. The average error distance is 55.05 cm, which is not accurate enough for sub-meter localization.

### 3.2.2 Equilateral Triangular Configuration



(a) 3 microphones are placed at vertices of a 20cm equilateral triangle. The average error is 18.6 cm  
 (b) 3 microphones are placed at vertices of a 40cm equilateral triangle. The average error is 10.04 cm

Fig 3.5a shows the accuracy when microphones are placed at the three vertices of a 20 cm equilateral triangle. The region inside the array has good accuracy. However, for regions along the line connecting any two microphones, the accuracy drops significantly. The average distance error across the region is 18.6 cm.

To evaluate the array size's impact on accuracy, the size of the original array (as in fig 3.5a) is increased by a factor of 2. The result is presented in fig 3.5b. The overall uncertainty area decreased across the region. The average error distance improved to 10.04 cm. By comparing with the previous result, it shows that increasing array size is effective in improving the overall accuracy.

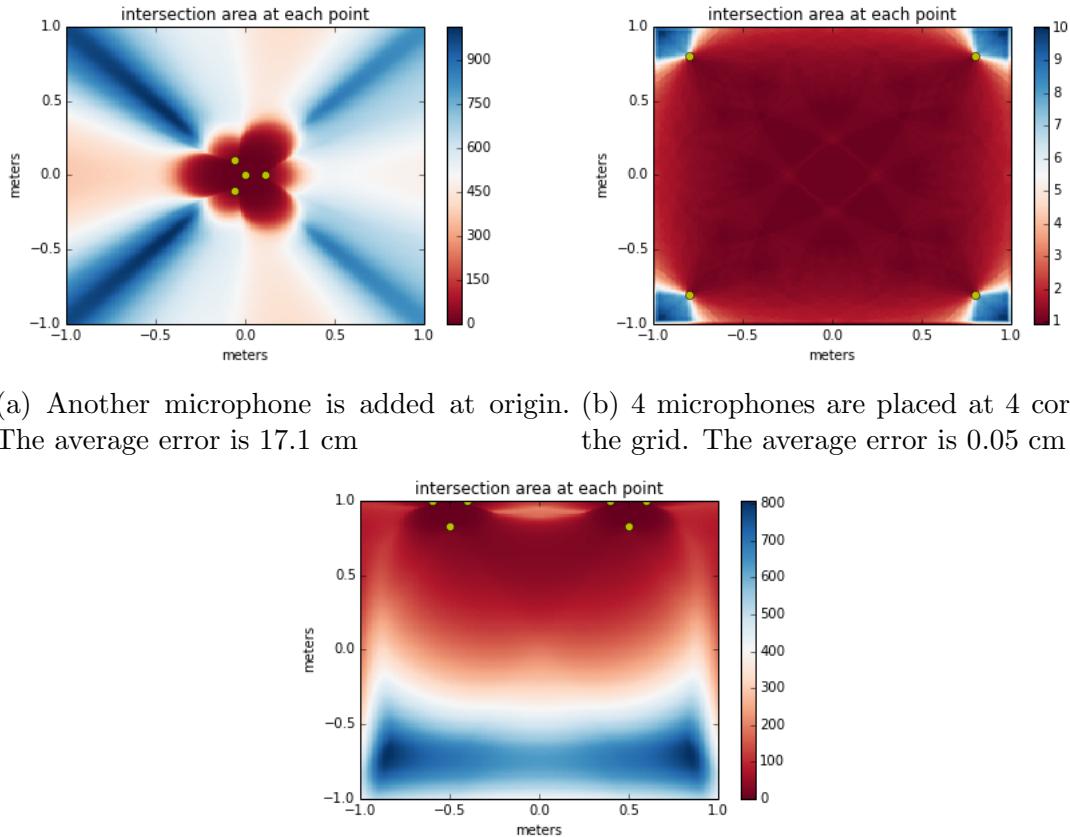


Figure 3.6: Error heatmap for different array configurations. The heatmap scale is the intersection area measured in  $cm^2$

### 3.3 More than three

To evaluate how adding one microphone (without increasing the array size) improves accuracy, another microphone is added at  $(0,0)$  for the array described in fig ???. Result is presented in fig 3.6a. Addition of the new microphone only slightly improved the accuracy around the array region. The average distance error dropped from 18.6 cm to 17.1 cm. Regions near lines connecting microphones still have significantly large intersection area. This result suggests that adding more microphones in a small microphone array (without increasing the array size) is not an effective method for improving the localization accuracy.

To further increase the distance between microphones, we placed four microphones at four corners of the region. Fig 3.6b shows the result. With this configuration, accuracy is consistently good across the region. The average distance error is 0.05 cm. However, placing microphones far apart at corners of the region requires accurate placement of all four individual microphones. The system is less portable compared to small arrays with microphones near each other. Placing microphones far apart from each other also causes problems in TDOA estimation, because sampling of microphones in the same array requires synchronized clock.

To avoid the need to accurately place microphones at far distances (as required by fig 3.6b), we explored configuration with two arrays. Two 3 microphone array are placed 1 meter apart and the result is presented in fig 3.6c. The result indicates that this configuration has good accuracy when source is close to the arrays. We observe that the localization accuracy decreases as the sound source moves outside of the region one meter by one meter away from the two arrays. The average error is 2.60 cm.

## 3.4 Discussion

Looking at the simulation results on localization accuracy for different microphone array size and configurations, we see that the further away the microphones are placed from each other the higher the localization accuracy. However, as we increase the distance between each microphone pairs, we size of the overall array increases and the system becomes less portable. In the end we decided to build the two array system as described in fig 3.6c. The setup is reasonably portable (compared to fig 3.6b), yet it can achieve on average high localization accuracy within a region one meter by one meter close the setup. Since our target application is HCI, an one meter by one meter region will be adequate for this purpose and it will be used for all the following experiments.

# Chapter 4

## Experiment

### 4.1 System

The end system has two arrays, each with three microphones mounted on the vertices of a 20 cm equilateral triangle. We used omni-directional foil electret condenser microphones due to their low cost. The microphone has a frequency range of 100 to 10K Hz, and a minimum SNR of 58dB[x]. We used operational amplifier OPA344 from Texas Instrument to amplify the microphone output by a factor of 100. A micro-controller is attached to one of the vertices. Fig 4.1 shows a picture of the array setup. The micro-controller baord in this project is *teensy 3.1*. It contains a which has a onboard RAM of 64k. ADC is capable of sampling at 500kHz [16, 17]. In this project, the micro-controller collects microphone data on all three channels for 12 milliseconds and then sends the recorded data to a computer through the USB port for localization.

To handle the uncertainty in TDOA estimation, instead of using point estimate that maximizes equation 2.9, we take the cross-correlation output(equation 2.10) as a measure of the likelihood of different arrival time differences. For each microphone array, we build a heatmap for the region. The intensity at each point on the heatmap represents the likelihood of it being the source. To generate the heatmap for an microphone array,

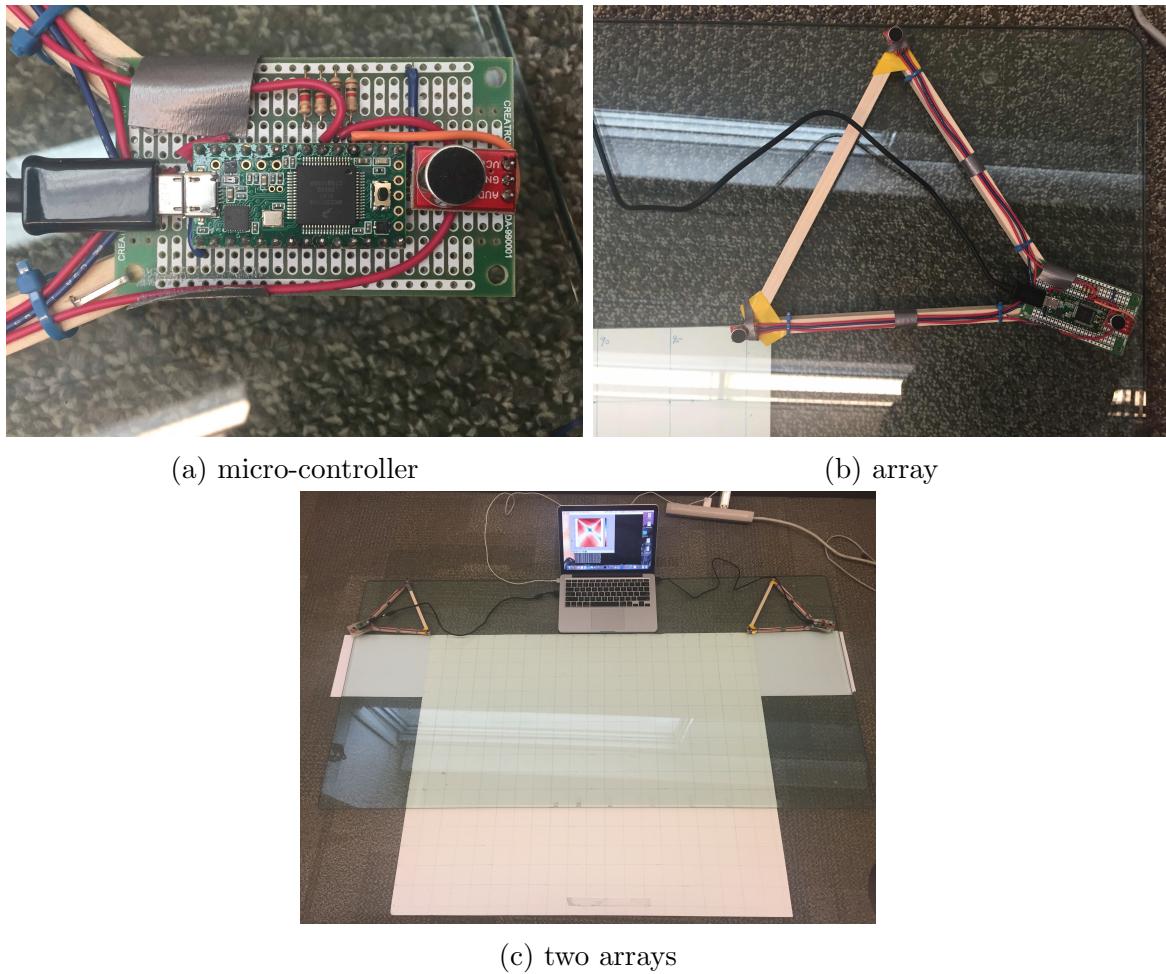


Figure 4.1: Localization system setup

we apply the following algorithm. For each point on the grid, the theoretical TDOA to each microphone pair can be precomputed. Then the heatmap can be generated by going through points on the grid and performing a lookup using equation 2.10. With three microphones  $m_1$ ,  $m_2$ , and  $m_3$ , there are three microphone pairs:  $m_1m_2$ ,  $m_1m_3$ , and  $m_2m_3$ . The theoretical TDOA from each location  $(x, y)$  to each microphone pair is precomputed and stored in  $D_{m_1,m_2}(x, y)$ ,  $D_{m_1,m_3}(x, y)$ , and  $D_{m_2,m_3}(x, y)$ . Then the likelihood map  $L(x, y)$  can be built by superposing the likelihood from each microphone pair:

$$L(x, y) = R_{m_1, m_2}(D_{m_1, m_2}(x, y)) + R_{m_1, m_3}(D_{m_1, m_3}(x, y)) \\ + R_{m_2, m_3}(D_{m_2, m_3}(x, y))$$

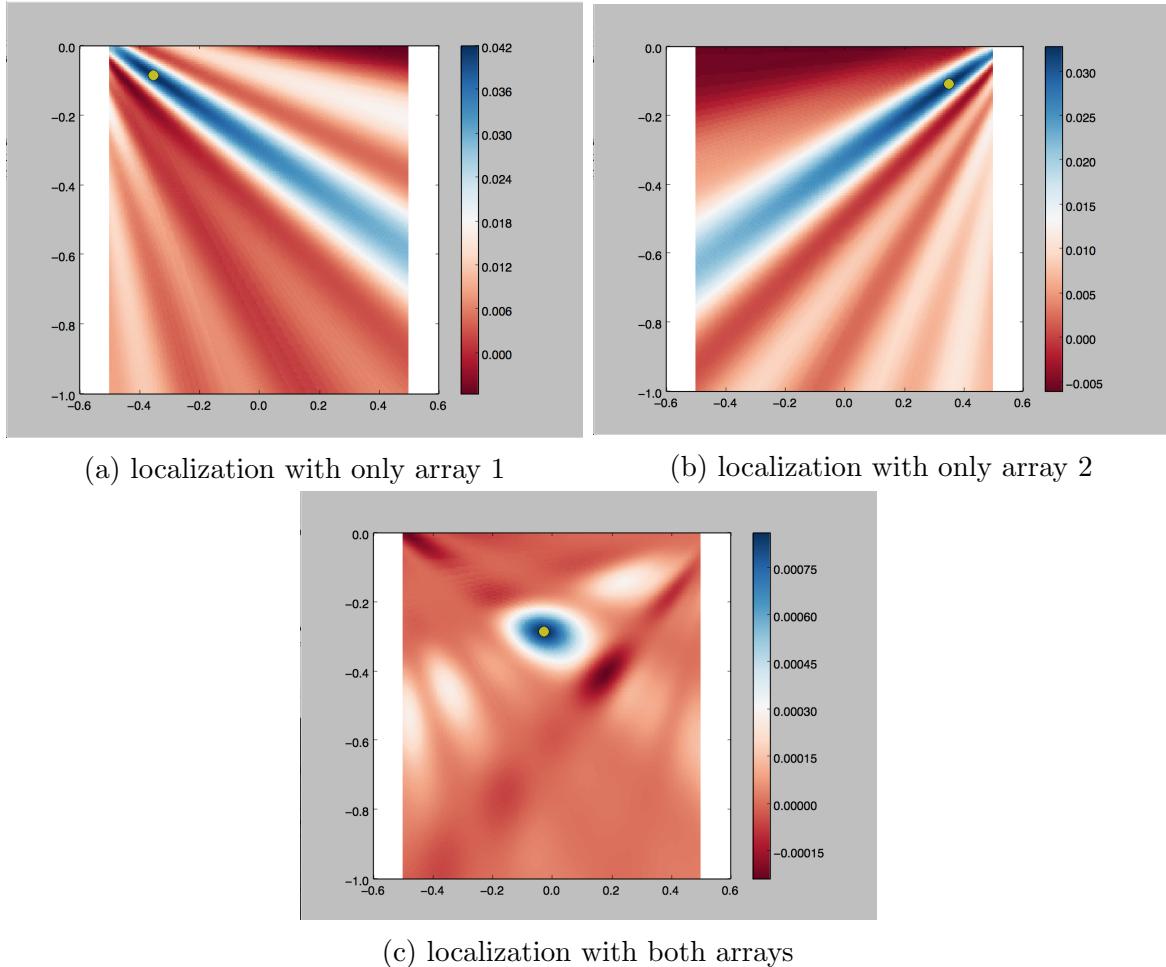


Figure 4.2: Likelihood maps for localization. The source is placed at  $(0.0, -0.3)$  m

where  $R_{m_1, m_2}(\tau)$ ,  $R_{m_1, m_3}(\tau)$ , and  $R_{m_2, m_3}(\tau)$  denote GCC output from microphone pairs  $m_1 m_2$ ,  $m_1 m_3$ , and  $m_2 m_3$ .

Likelihood maps from two arrays can be combined into the final likelihood map:

$$L(x, y) = L_1(x, y)L_2(x, y) \quad (4.1)$$

where  $L_1(x, y)$  and  $L_2(x, y)$  represent the likelihood map from array 1 and array 2.

To see the effect of accuracy improvement using multiple arrays, fig 4.2 shows a real life localization where the source is placed at (0 cm, -30 cm). It shows the individual likelihood map from each array and also the combined likelihood map according to equa-

tion 4.1. Individual arrays give accurate angle estimate, but have high uncertainty in distance estimate. The combined likelihood map demonstrated that by merging estimates from two arrays the system is able to perform more accurate localization.

From a timing point of view, the micro-controller spends 12 milliseconds on sampling the microphone data before sending it to a computer for processing. Sending the data through the USB port takes another 15 milliseconds, and processing on the computer takes around 50 milliseconds. Therefore, the total time lag between sound source and localization is around 80 milliseconds.

## 4.2 Setup

We conducted two sets of experiments to evaluate the system's localization accuracy: one on point localization and the other on movement tracking. For the point localization experiment we looked into using different window size of audio signals and different prefiltering methods. The size of the window limits how far apart the microphone arrays can be. If the time delay from one microphone array to another exceeds the size of the window, the location of the source can never be estimated. A large window size gives a more complete acoustic signal which makes cross-correlation less prone to noise. However, the larger the window size the more time it takes to compute the cross-correlation. This is a trade off that we will address.

For the movement tracking experiment, on top of the varying window size, we also looked into varying the movement speed of the audio source, applying different movement filters, and using different types of audio source. By applying an increasing movement speed of the audio source, we can test how fast the system can track an audible object moving in real time. We experimented with two types of music as our audio source for movement tracking, one with no low volume throughout, and the other with intermittent low or no volume. We want to test how the system performs when the audio source isn't continuously outputting sound. Furthermore, we applied different movement filters to help reduce noise and smooth out the path of the moving audio source.

### 4.2.1 Point localization

An one meter by one meter grid was set up where the arrays were placed at the top left and top right corners of the grid. Fig 4.3 shows a picture of the setup. A total of 32 testing locations were chosen uniformly in this grid. Testing is done by placing the audio source at each grid location, and turning on the audio source with white noise. We reported the error as the average distance between our placement of the audio source

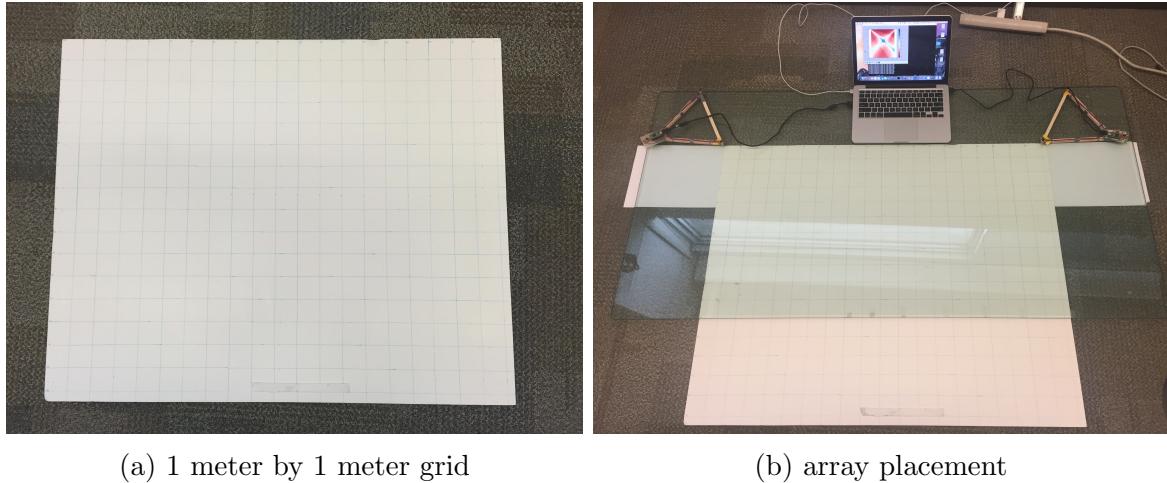


Figure 4.3: Setup for point localization evaluation

and the location estimated from the arrays.

#### 4.2.2 Movement tracking

To test how well the system tracks movement, we mounted a rotating disk 40 centimeters in diameter onto the grid at ( $x = 0$  cm,  $y = -30$  cm). Fig 4.4 shows a picture of the setup. A sound source is placed on the edge of the rotating disk and the arrays track the sound source as it rotates in a circle. In this experiment we used GCC\_PHAT as the prefiltering for cross-correlation because in the point localization experiment we found out GCC\_PHAT gives the best result. In this experiment, we evaluated how accuracy changes with:

- window sizes
- audio sources
- movement tracking filters
- movement speeds

To test how localization accuracy varies with different window size of audio signal received, we conducted the experiment with audio signals with varying length from 1.02

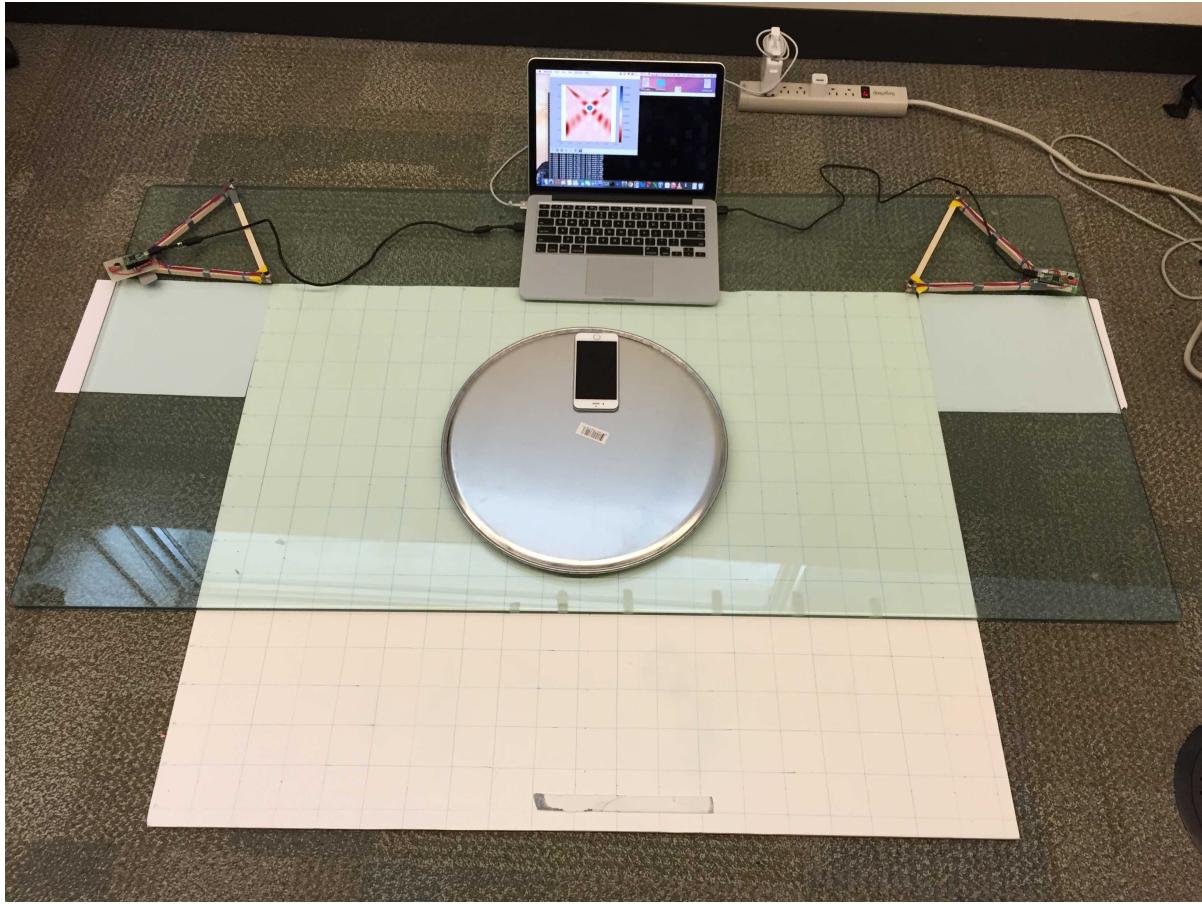


Figure 4.4: Setup for movement tracking evaluation

ms to 12 ms.

To test how different sound sources impact localization quality, we conducted the experiments with three different sound sources:

**White Noise** A recording of white noise. We expect GCC\_PHA<sup>T</sup> works best with white noise. The white noise is generated by sampling uniform randomly between  $-1$  and  $1$ .

**Music A** A randomly picked music that has non-zero audio amplitude throughout the experiment period. *Honest Eyes* by *Black Tide* was the music used.

**Music B** A randomly picked music with intermittent low amplitude sections. *Canon* was the music used.

To test how the movement speed of sound source affects localization quality, each experiment was conducted at two different speeds:

**Normal** An angular speed of 0.5 rad/s was maintained, which translates to a linear speed of 10 cm/s.

**Fast** An angular speed of 1.0 rad/s was maintained, which translates to a linear speed of 20 cm/s.

For each experiment conducted, two different movement filters were evaluated:

**Averaging filter** localization for past 0.5 seconds were averaged and outputted as current estimate.

**Kalman filter** A 2nd order Kalman filter was used.

In the movement tracking experiments described above, we have the ground truth location of the circle, but not the exact location of the audio source at each moment during the movement. Therefore, the error is reported as the distance between the localized point to its closest point on the ground truth circle.

## 4.3 Results

### 4.3.1 Point localization

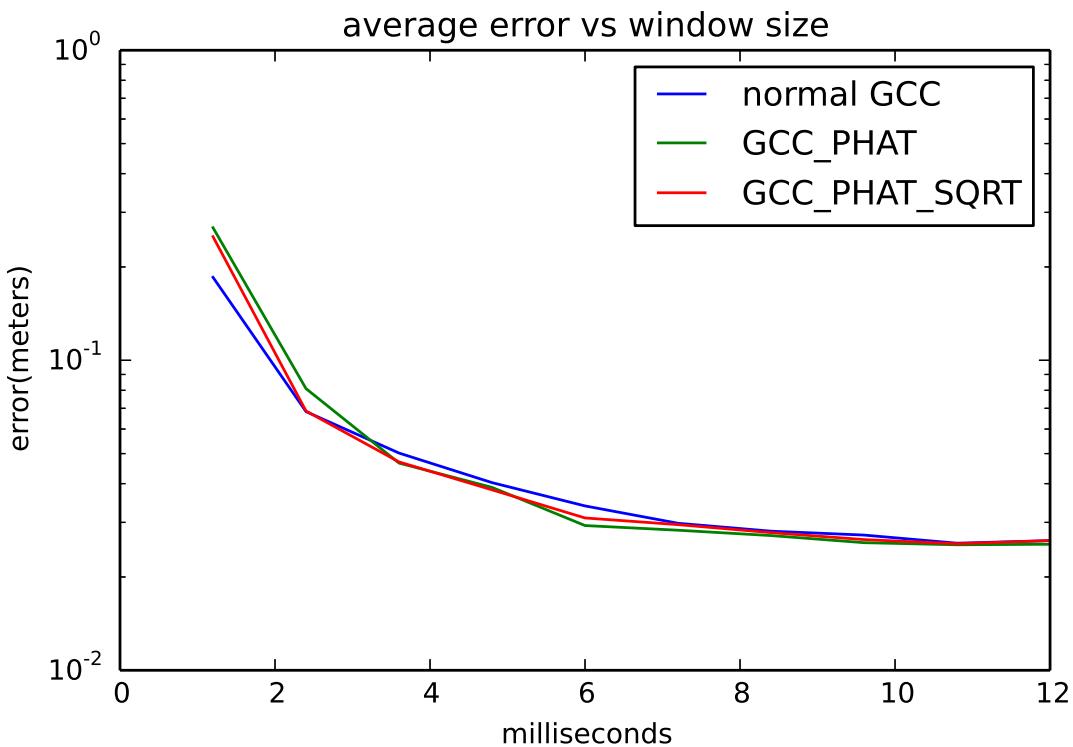


Figure 4.5: Localization error versus window size

To test how accuracy varies with window size, the algorithm is fed with microphone data of different lengths, and the result is shown in fig 4.5. The error decreases as window size increases and plateaus after the window size exceeds around 10 millisecond. The lowest error achieved is 2.53 centimeters, which occurred when the window size is set to 12 millisecond and GCC\_PHAT is used for TDOA estimation. The performance differences among GCC, GCC\_PHAT and GCC\_PHAT\_SQRT is small.

As was mentioned before, although accuracy improves with the window size, computation time also increases with it. The part of calculation that depends on the window size is using cross correlation for TDOA estimation. Cross correlation can be calculated with Fast Fourier Transform (FFT) and the runtime is of order  $O(N \log N)$ . We mea-

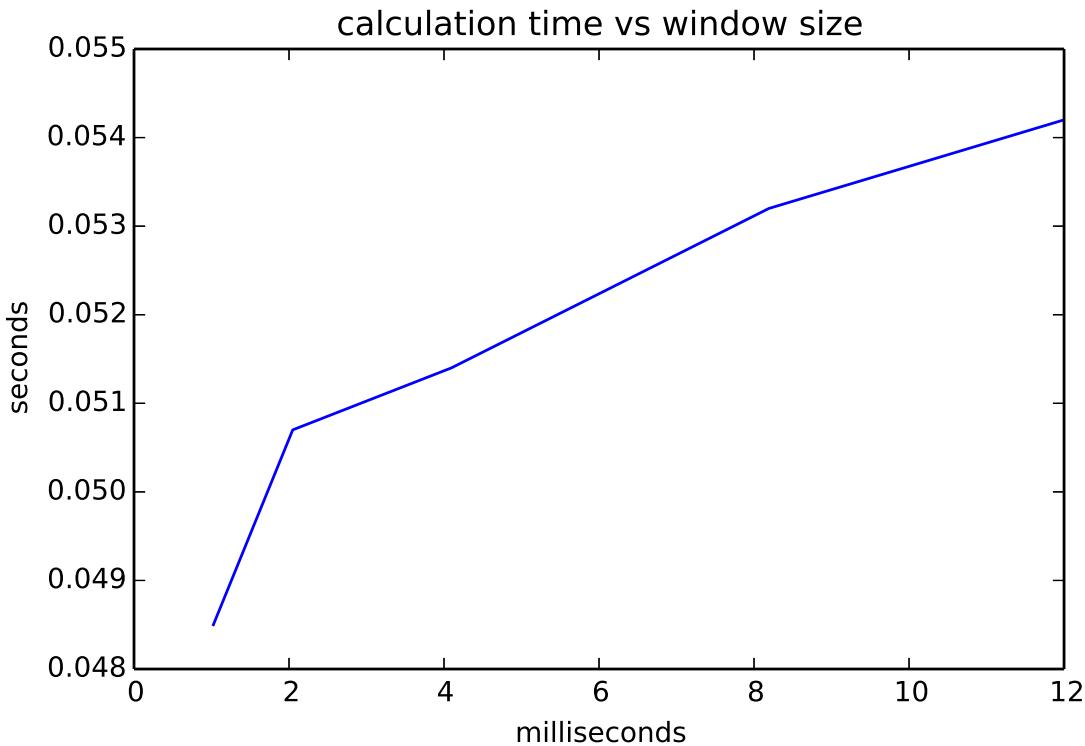


Figure 4.6: Computation time versus window size

sured how the computation time varied with window size and Figure 4.6 shows the result. The runtime increases approximately linearly in the window size region of interest.

We also calculated the localization error for each tested point in the grid. Figure 4.7 shows the error distribution inside the grid. The error is below 3 cm for most areas inside the grid. There is one error spike in the mid-left region and we contribute this to audio source misplacement because the error is fairly low and consistent around that spike.

To test the limit of the system and to evaluate the accuracy when the source moves outside of the one meter by one meter region, we measured the localization error by placing the source at 10 locations along  $y$  axis ranging from  $(0 \text{ cm}, -10 \text{ cm})$  to  $(0 \text{ cm}, -200 \text{ cm})$ . The result is presented in fig 4.8. The localization error is within 3 cm when the source is within 100 cm from the arrays. The error increases to about 5 cm when the source distance increases to 150 cm and the error exceeds 10 cm after the source distance reaches 200 cm.

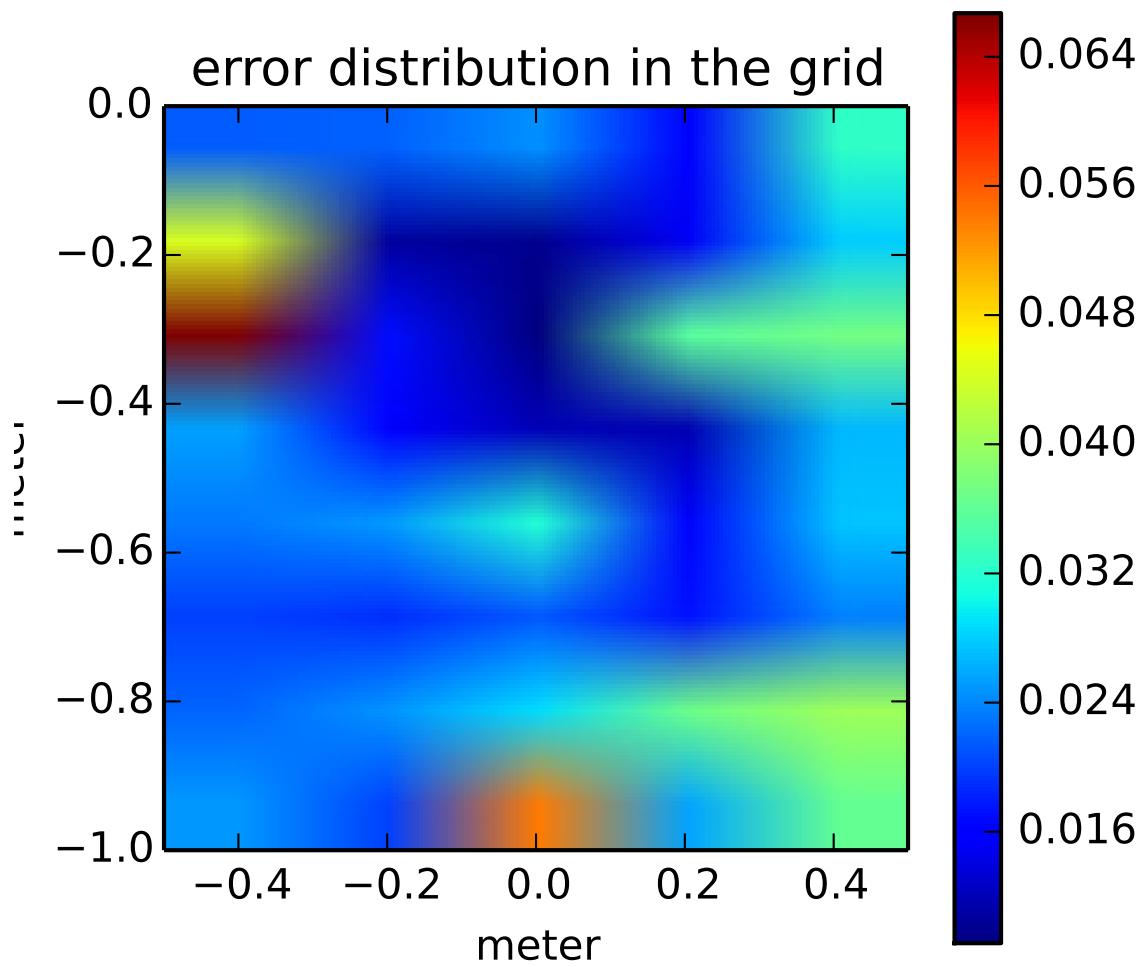


Figure 4.7: error distribution in the grid. Arrays are placed at  $(-0.5 \text{ m}, 0 \text{ m})$  and  $(0.5 \text{ m}, 0 \text{ m})$

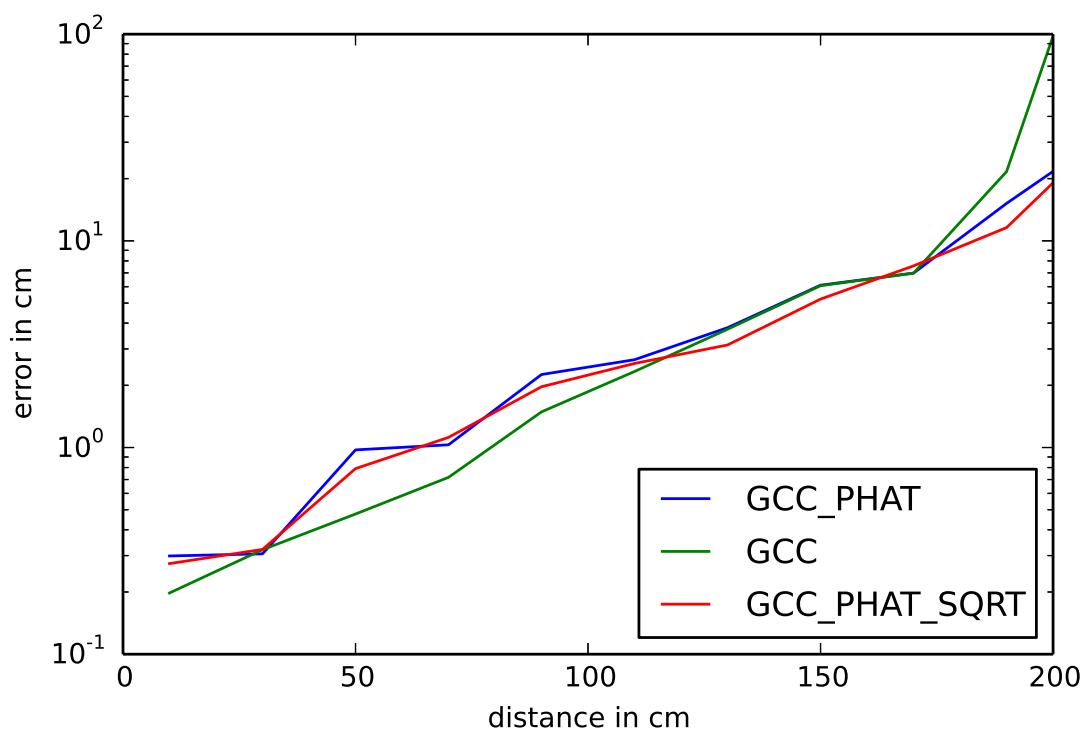


Figure 4.8: Localization error as the distance between the source and the microphone arrays increases. The source is placed on the  $y$  axis.

### 4.3.2 Movement tracking

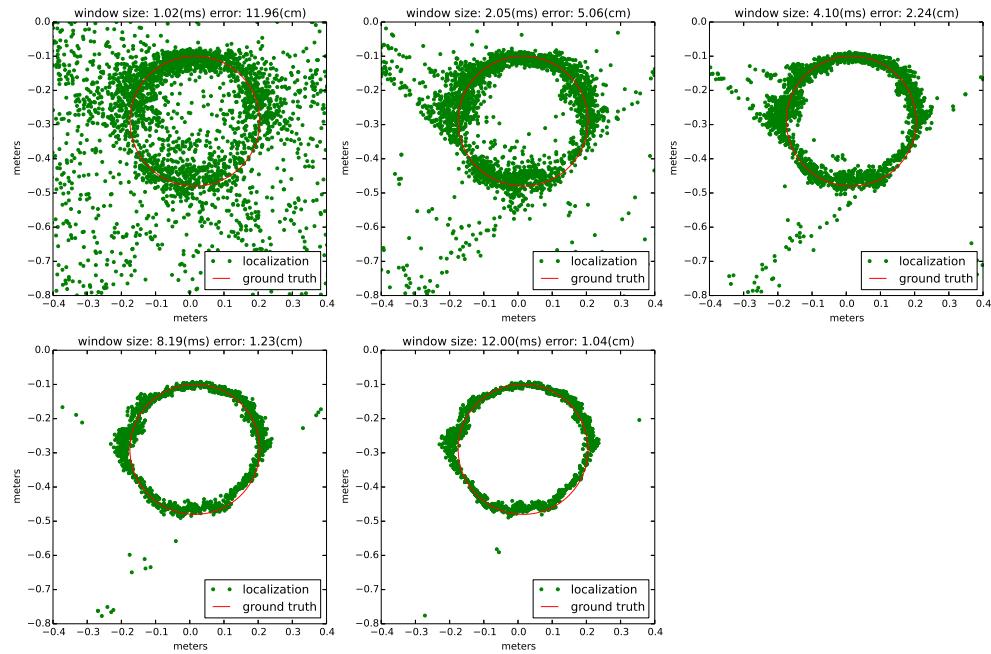


Figure 4.9: Localization quality versus window size

Fig 4.9 gives an intuitive representation of how accuracy changes with window size. When window size is small (1.02 millisecond), the audio does not contain enough information to reliably estimate TDOA, which results in noisy localization. As window size increases, the TDOA estimation becomes more accurate and the localization converges to the shape of the ground truth circle. Fig 4.10 shows how the error changes with window size. The general trend is similar to that in point localization case. Error decreases as the window size increases and plateaus after it exceeds around 10 milliseconds.

Fig 4.11 shows the result when white noise is used as the sound source and the source is rotated at 10 cm per second. The top right plot in this figures shows the raw detection output with the ground truth circle overlayed on top. It shows that the array's raw output matches the underneath ground truth circle reasonably well. The average error between the localization output with its closest point on the ground truth circle is 0.9



Figure 4.10: Localization error versus window size

cm. The top left plot in the figure shows the average error with different movement filters applied as a function of window size. The error decreases as window size increases and until the window size exceeds around 10 ms. When the window size is 12 cm, the localization accuracy is similar among raw output, Kalman filtering and averaging filtering. The bottom left plot shows the tracked movement for Averaging filter and the bottom right filter shows the tracked movement for Kalman filtering. Kalman filtering result is smoother while the averaging tracking stays closer to the ground truth circle. Compared to the experiment in fig 4.14, where the same sound source is played but at a faster rotating speed of 20 cm per second, we find that the performance is equally well between these two movement speeds.

Fig ?? shows the result when Music A is used as the sound source. The top right plot shows that the array still tracks the movement well, but has a slightly larger deviation compared to that in fig 4.11. The average localization error for Music A is 1.289 cm.

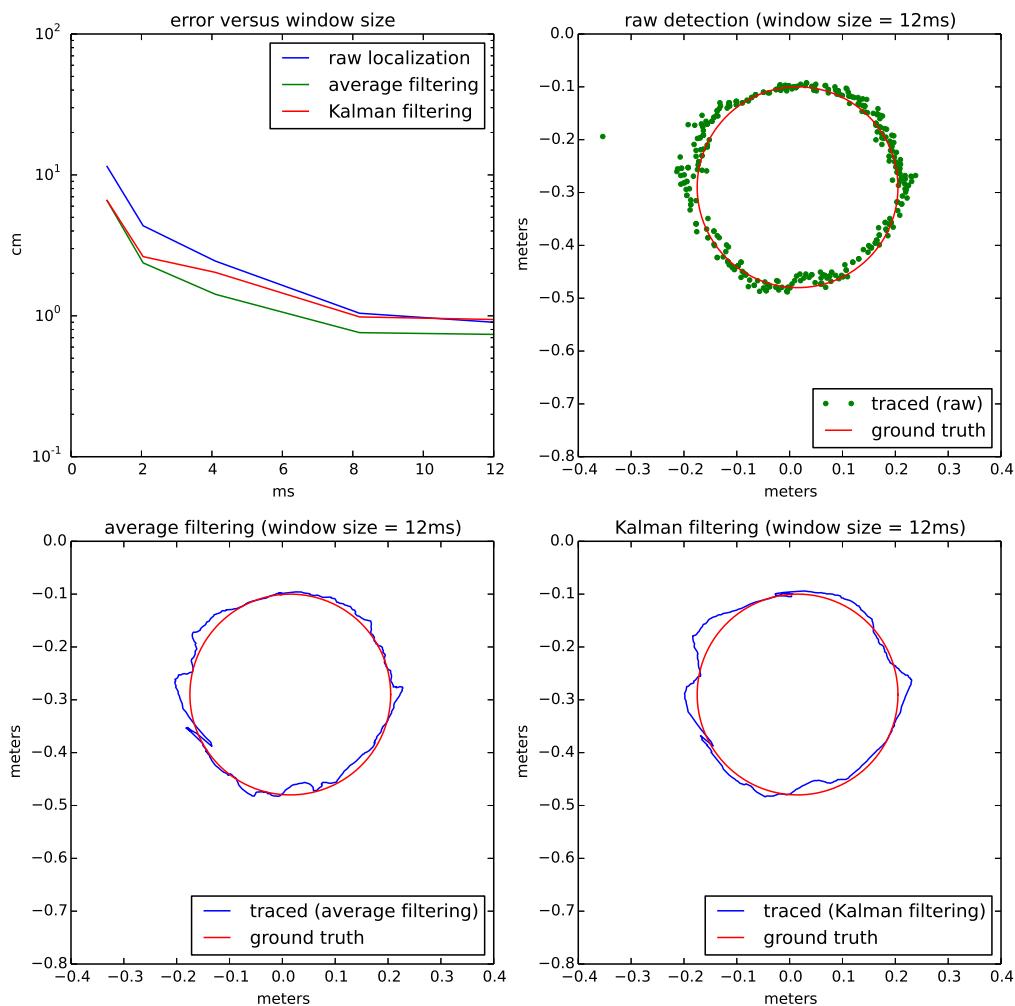


Figure 4.11: white noise (10 cm per second)

From the top left plot, it can be seen that the error still decreases with the window size. The performance improvement from raw output to Kalman filtering output is bigger than that with white noise (top left plot of fig ??). Averaging filtering still produces lowest average error. Fig ?? shows the result for the same sound source moved at twice the speed (20 cm per second). The average localization error at the faster speed is 1.291 cm. Compared to the experiment where the same sound source is placed but with a

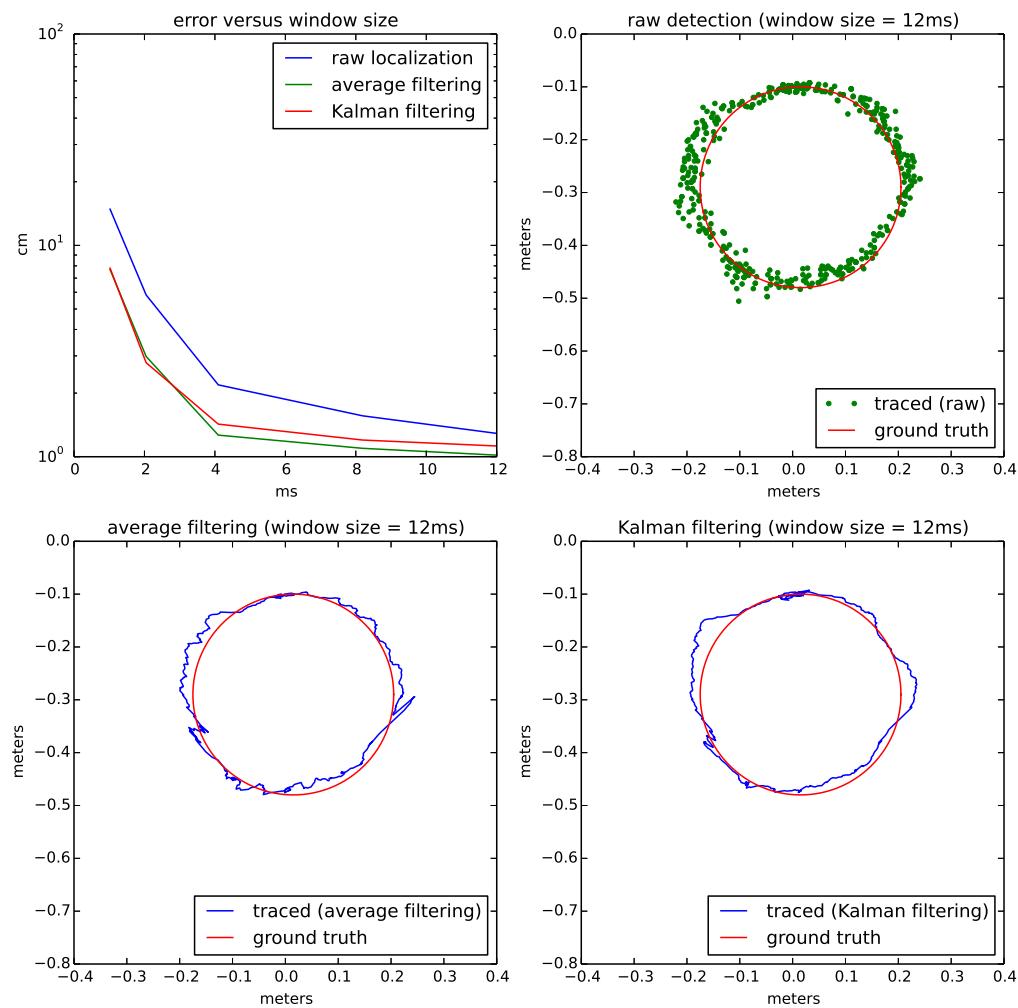


Figure 4.12: music A (10 cm per second)

faster movement speed (fig ??, we found that the performance does not degrade as the movement speed increases from 10 cm per second to 20 cm per second).

Fig ?? shows the result when Music B is used as the sound source. The low amplitude intervals in Music B affects the performance significantly. The average localization error is 2.9. The “blank” regions in the music can also be visually seen from the top right plot. The top left plot shows that the Kalman filtering still performs better than of the white

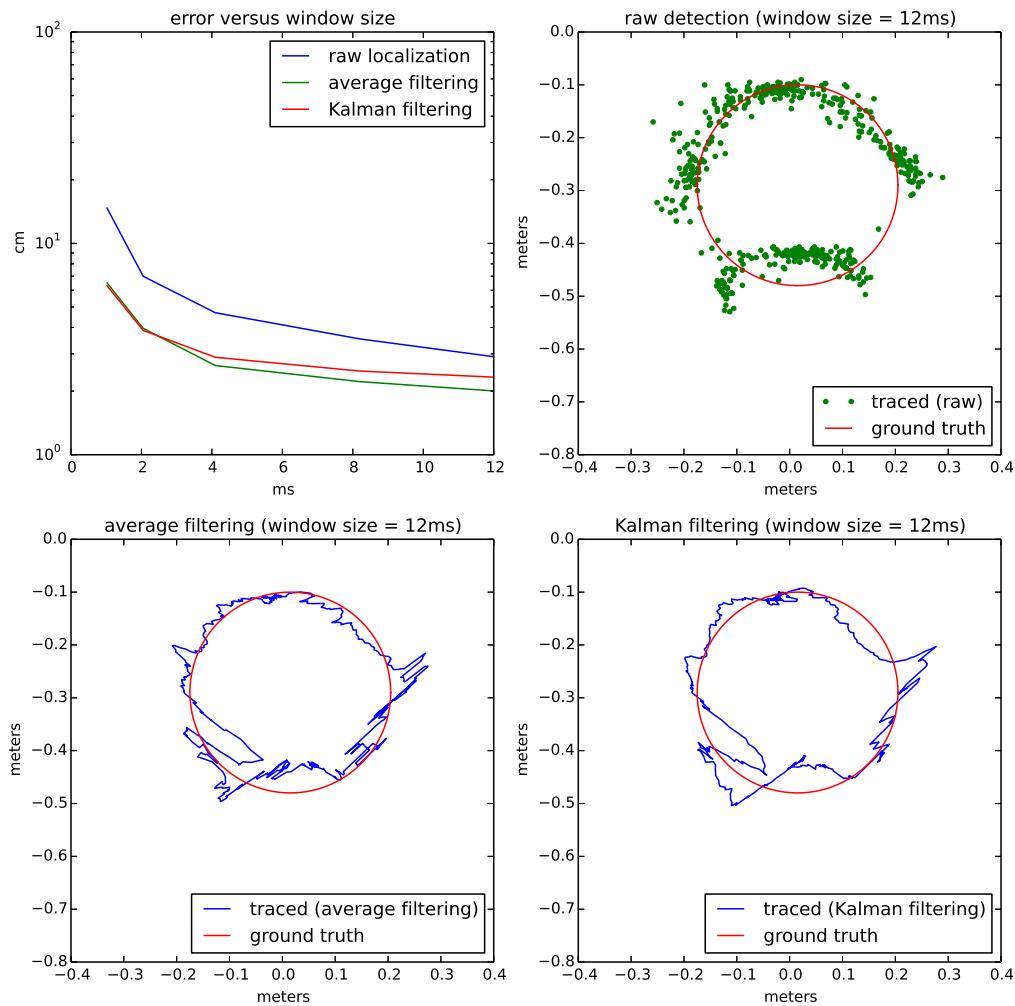


Figure 4.13: music B (10 cm per second)

noise, and behaves similar to the case with Music A. Averaging filtering produces lowest average error. Compared to the faster movement experiment of the same sound source, we find that the localization error is similar between these two movement speeds.

Fig 4.11 to 4.13 shows results for experiments at normal speed, and Fig 4.14 to 4.16 shows results at fast speed. By comparing localization error for each audio source between normal movement speed and fast movement speed, we find that localization error does not

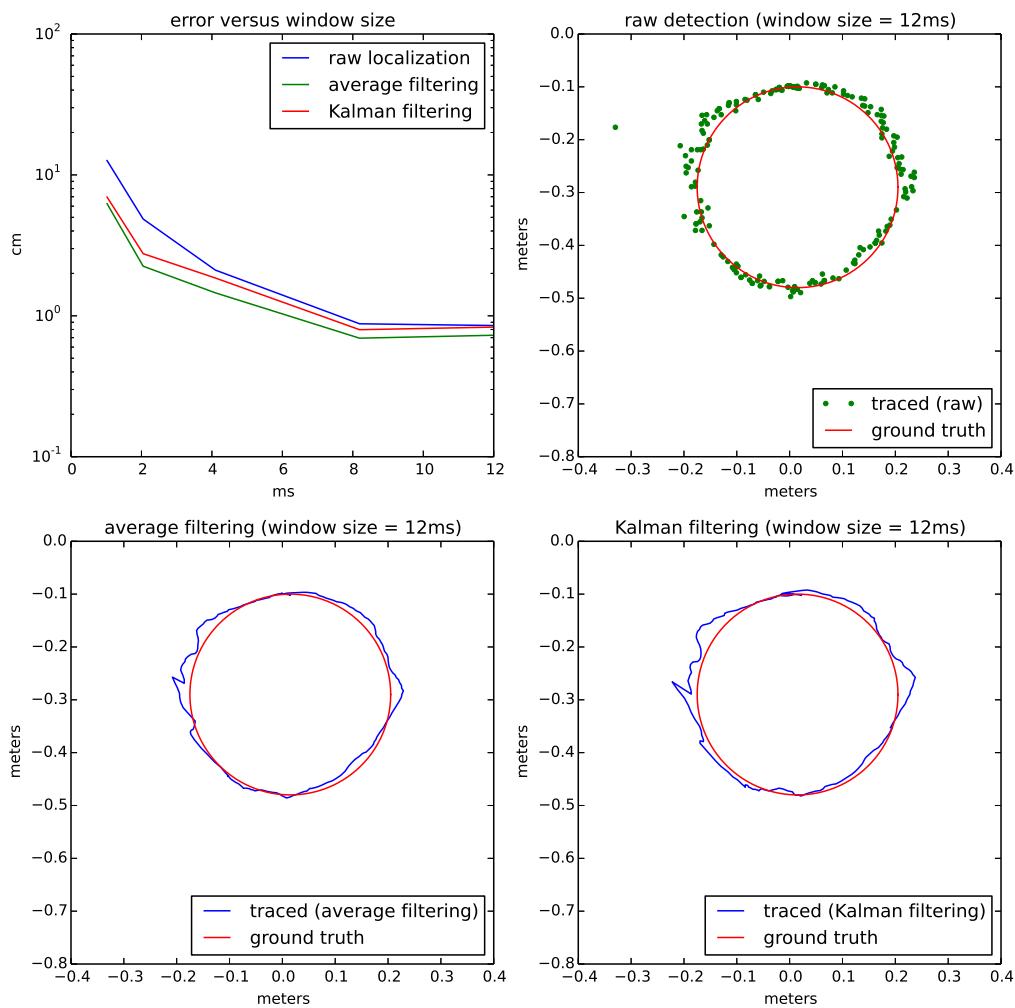


Figure 4.14: white noise (20 cm per second)

increase when the source movement speed increased from 10 cm to 20 cm. For example fig 4.12 and fig 4.15 shows that localization error is 1.289 cm at normal movement speed and 1.291 cm at fast movement speed.

For normal speed movement tracking, localization error is 0.9 cm for white noise (Fig 4.11), 1.29 cm for Music A (Fig 4.12), and 2.9 cm for Music B (Fig 4.13). Localization accuracy is the best for white noise source, and worst for Music B. This is consistent with

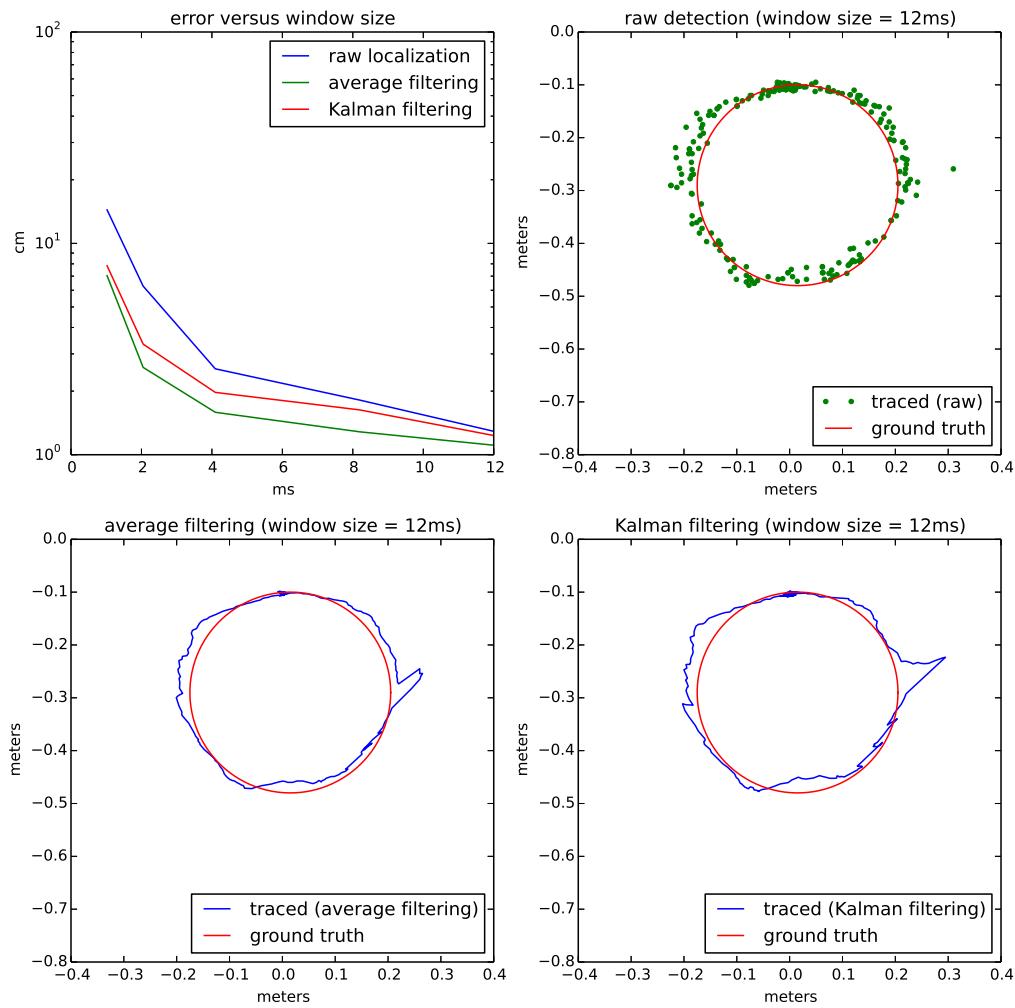


Figure 4.15: music A (20 cm per second)

our expectation because low amplitude regions in Music B would cause arrays to lose track of where the source is. This can also be seen from the “blank” regions in fig 4.13.

### 4.3.3 Discussion

It also shows that raw detection has the most amount of jiggling. Kalman filter reduces the amount of jiggling from raw detection by combining past estimates with current

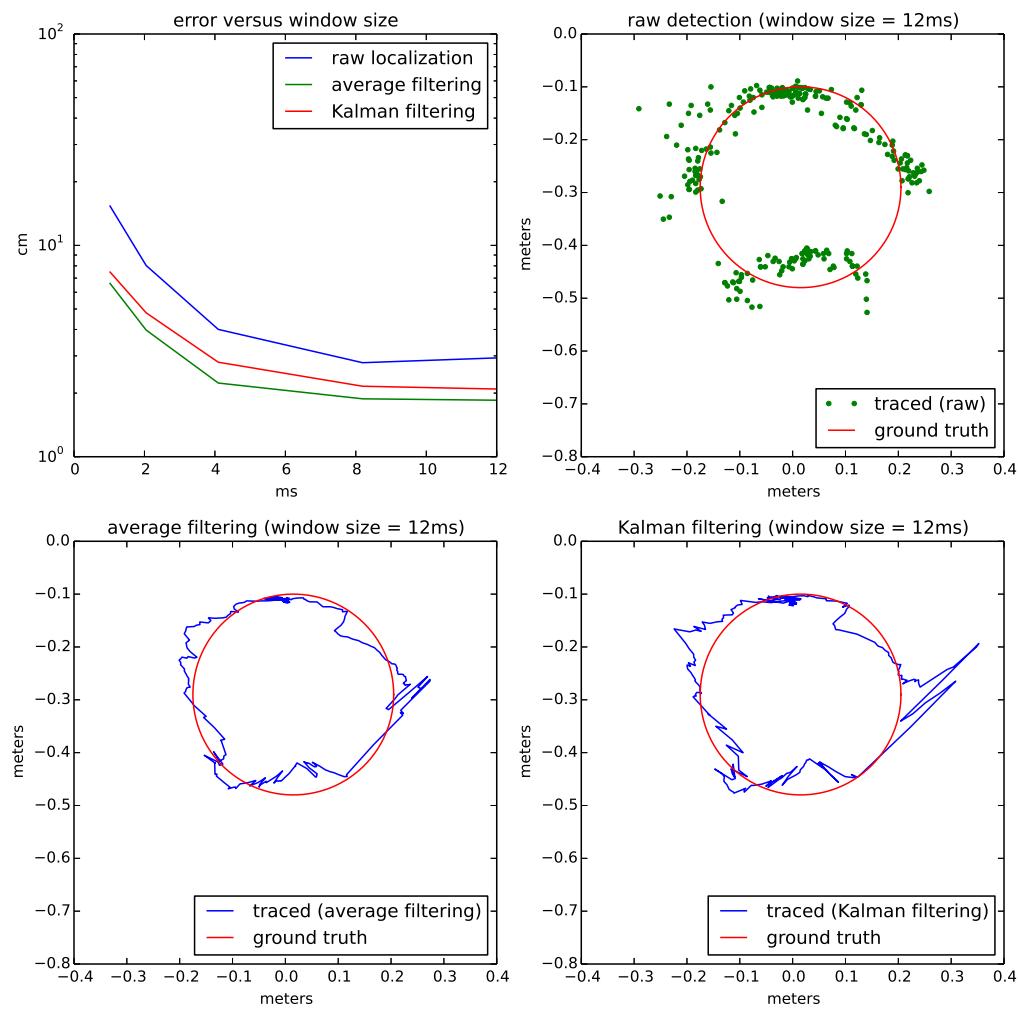


Figure 4.16: music B (20 cm per second)

estimates. Averaging filter has the least amount of jiggling. However, averaging filter averages detection outputs from past 0.5 seconds, which makes the filtered output lag the real movement.

# Chapter 5

## Conclusion

In this thesis, we described and built an inexpensive, portable yet reasonably accurate sound localization system with two microphone arrays. We analyzed different array architectures and showed that the localization accuracy varies with the array geometry and the location of the sound source (both the distance and the angle). In particular, regions close to the line connecting two microphones are generally more difficult to localize than regions close to the line bisecting two microphones. This problem can be alleviated by employing multiple microphones that are placed at large distance from each other, but such arrangement affects the portability of the system. As a trade off between accuracy and portability, we demonstrated that a two array architecture achieves better accuracy compared to a single array system of similar size. By using cross-correlation output as the likelihood for different arrival time differences, each array generates a likelihood map for each possible location in the area. We demonstrated that by merging likelihood maps from the two arrays, we were able to achieve good localization accuracy (less than 3 cm average error) in a local one meter by one meter region. Since two arrays generate their likelihood maps individually, this system does not require synchronized clock between the arrays, which made the overall system easier to design and more portable.

This system can be used in HCI applications that incorporate sound position and

movement information. One can use this system to build virtual drawing applications where the user can draw with music, without physically touching the computer. One can also use this system to design interactive AI games. For example, a chess game with physical pieces can be developed where each piece is equipped with a motor and a music tag. Since the computer knows where all the pieces are, it knows which piece the user has moved and can make its corresponding move. A similar example is AI toy car racing game, where the player controls one car and the computer controls another car. With real time location information, the computer can control one car to compete with the player on a racing track. This system can also be used to in Augmented Reality (AR) applications. For example, users can wear a music tag on the finger, and then the system would be able to track finger movement. This particular setup can be used as a virtual mouse for wearable technologies such as Google Glass.

For future directions, one can extend the system with another microphone on each array to perform 3D localization. However, the number of grid points increases exponentially with the number of dimensions, which translates into exponential increase in computation time. To keep the localization in real time (time lag of less than 0.1 seconds), it is important to research more efficient ways of finding the intersection of hyperbolic cones the in the 3D space. Since the likelihood calculation on each grid point is simply a look up on the cross-correlation output from each microphone pair, the computation at each grid point is independent of each other. Therefore, One straightforward way to speedup calculation is to employ hardware accelerated parallel computing technologies such as GPU or FPGA.

Another direction is to investigate multi-source localization. One approach is to allocate different frequency bands to difference sound sources. However, the number of frequency bands are limited in the audible audio spectrum, and our results showed that TDOA estimation is more accurate for a wide band signal compared to a narrow band signal. A different approach is to digitally encode and tag the audio source. Different

sources can be differentiated if each source uses a waveform that has a different digital signature.

# Bibliography

- [1] Wing, Michael G. and Eklund, Aaron and Kellogg, Loren D. “Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability,” *Journal of Forestry*, 2005.
- [2] Bekkelien, Anja, Michel Deriaz, and Stphane Marchand-Maillet. “Bluetooth indoor positioning,” *Master’s thesis, University of Geneva*, 2012.
- [3] Hui Liu; Darabi, H.; Banerjee, P.; Jing Liu, “Survey of Wireless Indoor Positioning Techniques and Systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2007
- [4] Yazici, A.; Yayan, U.; Yucel, H., “An ultrasonic based indoor positioning system,” *2011 International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2011
- [5] Birchfield, Stanley T., and Rajitha Gangishetty, “Acoustic localization by interaural level difference,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005
- [6] Blauert, Jens. “Spatial hearing: the psychophysics of human sound localization,” *MIT press*, 1997.

- [7] Raspaud, Martin, Harald Viste, and Gianpaolo Evangelista, “Binaural source localization by joint estimation of ILD and ITD,” *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.
  - [8] Guentchev, Kamen, and John Weng. “Learning-based three dimensional sound localization using a compact non-coplanar array of microphones,” *Proc. AAAI Spring Symposium on Intelligent Environments*, 1998.
  - [9] Chowdhury, T., Aarabi, P., Weijian Zhou, Yuan Zhonglin, and Kai Zou, “Extended touch user interfaces,” *IEEE International Conference on Multimedia and Expo (ICME)*, 2013.
  - [10] Pham, D. T., et al. “Tangible acoustic interface approaches,” *Proceedings of IPROMS 2005 Virtual Conference*, 2005.
  - [11] XueXin Yap, Khong, A.W.H., and Woon-Seng Gan, “Localization of acoustic source on solids: A linear predictive coding based algorithm for location template matching,” *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2010
  - [12] Chowdhury, Tusi. “Single Microphone Tap Localization,” *University of Toronto*, 2013.
  - [13] Ishii, Hiroshi, et al. “PingPongPlus: design of an athletic-tangible interface for computer-supported cooperative play,” *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1999.
  - [14] Paradiso, Joseph A., et al. “Passive acoustic knock tracking for interactive windows,” *CHI’02 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2002.
- APA

- [15] Paradiso, Joseph A., et al. “Passive acoustic sensing for tracking knocks atop large interactive displays,” *Proceedings of IEEE. Vol. 1. IEEE*, 2002.
- [16] MicLoc, URL: <http://ruralhacker.blogspot.ca/p/micloc.html>
- [17] PJRC, URL: <https://www.pjrc.com/teensy/teensy31.html>
- [18] Polotti, Pietro, et al. “Acoustic Localization of Tactile Interactions for the Development of Novel Tangible Interfaces,” *Proc. of the 8th Int. Conference on Digital Audio Effects (DAFX-05)*, 2005.
- [19] Paradiso, Joseph A., et al. “Sensor systems for interactive surfaces,” *IBM Systems Journal*, 2000.
- [20] Checka, Nisha “A system for tracking and characterizing acoustic impacts on large interactive surfaces,” *Diss. Massachusetts Institute of Technology*, 2001.
- [21] Brandstein, M.S.; Silverman, H.F., “A robust method for speech signal time-delay estimation in reverberant rooms,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.
- [22] Knapp, C. and Carter, G.Clifford, “The generalized correlation method for estimation of time delay,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1976.
- [23] Aarabi, P. and Guangji Shi, “Phase-based dual-microphone robust speech enhancement,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2004.