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

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 achieves 12 localizations per second with an average error of less than 3 cm for both point localization and movement tracking in a local one meter by one meter region. Using our system, we have demonstrated painting with music as a direct application. Another potential application of our system is to track finger movements and perform gesture recognition. As a future direction, our system can be extended to localizing objects and tracking movement in 3D space.

Contents

List of Figures

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. Relevant commercial user case includes applications that serve advertisements and coupons to consumers in shopping centers based on the customer's location. In these systems, device signal strength received at different base stations are used for the estimation of the user device location. However, their reported accuracy are in the range of 1 to 5 meters [2, 3], which is not accurate 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. The purpose of our system is to give applications to designs that allow remote human computer interaction. To test the accuracy of our localization system, we perform acoustic localization experiments in an one meter by one meter area. The experiment area size is adequate for our purposes because human arm movement will generally be constrained in this area. Our experiment show that our system localizes both static and moving acoustic source with 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 artificial intelligence (AI) games with audible physical game 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 of both cars, 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 example, the user can wear a music tag on one finger, and then the system would be able to track the user's finger movements. This particular setup can be used as a virtual mouse for wearable technologies such as Google Glass.

1.1 Contribution

The main contributions of the thesis are:

1. We provided a detailed simulation analyzing the impact of the arriving time estimation accuracy on localization error.

2. We evaluated different array architectures, and showed that a two array architecture achieves good localization accuracy while maintaining good portability.

3. We implemented and built the proposed sound localization system and compared different state of the art methods for localization and tracking.

As a fun demonstration of the technology, we also demonstrated that our system can be used as a tool for painting in space.

1.2 Outline

We first discussed in Chapter II prior relevant research and approaches in sound localization. We also discussed various ways of combining past localization estimates for movement tracking. 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.

Chapter 2

Background

2.1 Point localization

Acoustic localization techniques have 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 when used to localize high frequency sources, because these sources don't diffract significantly 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:

$$I_i \propto \frac{1}{d_i^2} I_s \tag{2.1}$$

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:

$$I_1 d_1^2 = I_2 d_2^2 \tag{2.2}$$

$$\frac{I_1}{I_2} = \frac{d_2^2}{d_1^2} \tag{2.3}$$

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 they can not be distinguished from each other. Multiple approaches have been investigated to eliminate this ambiguity. The authors in [5] employed multiple microphones and used the intersection of circles from each microphone pair to estimate the source location. Authors in [7] combined ILD with Interaural Time Difference (ITD) to estimate the source direction (i.e azimuth). Instead of solving the intersection of circles, authors in [8] employed machine learning techniques to automatically learn the mapping from ILD and ITD features to location coordinate. Their technique 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 purposes, since in an interactive system we do

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.

Authors in [9] and [13] have investigated using max value from cross-correlation as a similarity measure to localize user taps on interactive surfaces. In [14], the authors used L2 distance in the Linear Predictive Coding coefficient space as a similarity measure to localize taps on surfaces. Authors in [15] 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. The authors in [15] have also investigated into 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 [20], 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 pre-set 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 [27] and [28] extracted descriptive parameters for each significant peak(e.g., peak height, width, mean arrival time). Their algorithm then used the extracted parameters to predict arrival time with a second order polynomial, the parameters of which were fitted at fixed locations during a calibration phase. Authors in [21] 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[22, 24, 26]. 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 τ can be calculated as:

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

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 τ 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 τ. With discrete signals of length n, it will take $O(n^2)$ number of calculations. Doing the same calculation in frequency domain, we first need to transform the signal into frequency domain, then multiply 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 particularly beneficial in real-time interactive systems since significant time lag would make the system less interactive to the user.

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 [29, 30, 31]. Under the GCC framework, the arrival time difference t_0 between two signals $x_1(t)$ and $x_2(t)$ is estimated as:

$$t_0 = \arg\max_{\tau} R_{x_1x_2}(\tau) \tag{2.9}$$

$$R_{x_1x_2}(\tau) = \int_{-\infty}^{\infty} W(\omega)X_1(\omega)X_2^*(\omega)e^{j\omega\tau}d\omega \tag{2.10}$$

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

GCC_PHAT_SQRT $W(\omega) = \frac{1}{\left|X_1(\omega)X_2^*(\omega)\right|^{0.5}}$. 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:

$$R_{x_1x_2}(\tau) = \int_{-\infty}^{\infty} W(\omega)|X_1(\omega)||X_2(\omega)|e^{j(\omega\tau-(\angle X_2(\omega)-\angle X_1(\omega))}d\omega \tag{2.11}$$

$$= \int_{-\infty}^{\infty} \underbrace{W(w)|X_1(\omega)||X_2(\omega)|}_{\text{weighting}} \cos(\Theta_\epsilon)d\omega \tag{2.12}$$

Where Θ_e is the phase error:

$$\Theta_\epsilon = \omega\tau - (\angle X_2(\omega) - \angle X_1(\omega))$$

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 τ is the true arrival time difference between $x_1(t)$ and $x_2(t)$, phase error $\Theta_\epsilon = 0$, and $\cos(\Theta_\epsilon) = 1$. When τ 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 $W(w)|X_1(\omega)||X_2(\omega)|$ 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 unfilterred GCC and GCC_PHAT.

2.2 Movement Tracking

In the previous section, we discussed various ways of localizing acoustic source. For each received microphone data, the algorithm produces a point estimate of the source location. If we have multiple estimates of the source location, they can be intelligently combined to make better location prediction.

In this section we describe two ways of combining past estimates to produce better estimates: Finite impulse response (FIR) filter and Kalman filter.

2.2.1 FIR filter

The impulse response is of finite duration for FIR filters. In our system, since only location estimates from the past are available, we look at a specific category of FIR filters: causal discrete-time FIR filters. In such systems, the output $y[n]$ is a linear weighted combination of past $N+1$ estimates from input $x[n]$:

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] && (2.13) \\ &= \sum_{i=0}^{N} b_i x[n-i] && (2.14) \\ & && (2.15) \end{aligned}$$

where

- $y[n]$ is the output sequence

- $x[n]$ is the input sequence

- N is the filter order

- b_i is the impulse response

By controlling the impulse response b_i, we specify how the past few data should be weighted to produce the desired output.

When $b_i = \frac{1}{N+1}$, each of the previous $N+1$ localization estimate contributes equally to the output. In this case, the filter becomes a simple averaging filter (also called rolling mean). If the source does not move during the past $N+1$ estimates, and assuming each estimate is an independent estimate of the true location with an additive Gaussian noise. Then it can be shown that the output after filter is an unbiased estimation of the true source location. However, if the source has moved during the past $N+1$ estimates, the filter would only output the mean location for the past $N+1$ estimated locations, which results in a "lagging" effect between the true source location and the system output. After the source has stopped moving, the filter output catches up with the source location.

To reduce the "lagging" effect exhibited by the averaging filter, we can assign higher weights to more recent estimates. Recent estimates gets a higher contribution to the output location, which makes the filtered output tracks more closely with the sound source. However, if the localization system is noisy with large estimation variance, then the error in the most recent estimate dominates the filtered output, which makes the filtered output prone to noise and exhibit large variance.

To overcome the "lagging" problem while making the system robust to noise, we can design a system that produce location estimates at a fast rate. This way we can make sure that the source would not have enough time to move a significant distance before the next estimate. Then it will be safe to average just a few estimates from the past.

2.2.2 Kalman filter

The Kalman filter is a recursive filter where input data can be efficiently combined to produce online prediction. If all noise are Gaussian, the Kalman filter is a statistically optimum filter that minimizes squared error of the estimated parameters[16, 18]. Even if the noise is not Gaussian, given only the first two statistics of noise, Kalman filter is the best linear estimator[17]. Due to its statistical optimality and its recursive nature that enables online prediction, the Kalman filter has found applications in a wide range of areas. It has been used to track aircraft using RADAR, and to track Robot with sensors and beacons[19].

Kalman filter uses observed variables to infer hidden variables and use them to help predict the next state. In our system, observed variable z is the (x, y) coordinates of the localized acoustic source:

$$z = \begin{bmatrix} x \\ y \end{bmatrix}$$

We can also model unobserved motion variables such as velocity $(\dot{x}, \dot{y})$ and acceleration $(\ddot{x}, \ddot{y})$. Then the state variables $\mathbf{x}$ that we are tracking can be represented as:

$$\mathbf{x} = [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]^T$$

Internally, the Kalman filter also keeps track of the uncertainty of the state variables. It is represented as a covariance matrix P on state variables.

Note that by modeling up to acceleration, we are implicitly assuming higher order motion variables are constant (such as jerk). This assumption gives the system a tracking bias if there is a jerk change. Kalman filter also includes a term Q that can be used to model the process noise.

At any time instant, Kalman filter can use current state information to infer the

predicted next state $\mathbf{x}^-$ and the predicted uncertainty P^-:

$$\mathbf{x}^- = F\mathbf{x} + Bu \tag{2.16}$$

$$P^- = FPF^T + Q \tag{2.17}$$

where

- F is the state transition matrix. In our system, next coordinates (x, y) can be computed with law of physics using current position, velocity and acceleration:

$$F = \begin{pmatrix} 1 & 0 & \delta t & 0 & \frac{1}{2}\delta t^2 & 0 \\ 0 & 1 & 0 & \delta t & 0 & \frac{1}{2}\delta t^2 \\ 0 & 0 & 1 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

  , where δt represents the time difference between current time and the time of last state update.

- The Kalman filter is a general framework which allows not only object tracking but also controlling the motion of the object. In these situations u is the control input and B describes the control input model. For our application, we are only tracking and not controlling the movement, $Bu = 0$.

- Q models the process noise

After the observation for the next coordinates are made, then the Kalman filter calculates the residue between the prediction $\mathbf{x}^-$ and the measurement z:

$$y = z - H\mathbf{x}^-$$

where H is the measurement function that transforms from state space to measurement space. In our example, the transformation is simply taking the location (x, y) from the state space:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Then the Kalman filter updates the states estimates $\mathbf{x}$ and states uncertainty estimates P:

$$\mathbf{x} = \mathbf{x}^- + Ky \tag{2.18}$$

$$P = (I - KH)P^- \tag{2.19}$$

where K is the Kalman gain:

$$K = P^- H^T S^{-1} \tag{2.20}$$

$$S = HP^- H^T + R \tag{2.21}$$

R is the measurement noise matrix that models the localization system's output noise as a covariance matrix.

Chapter 3

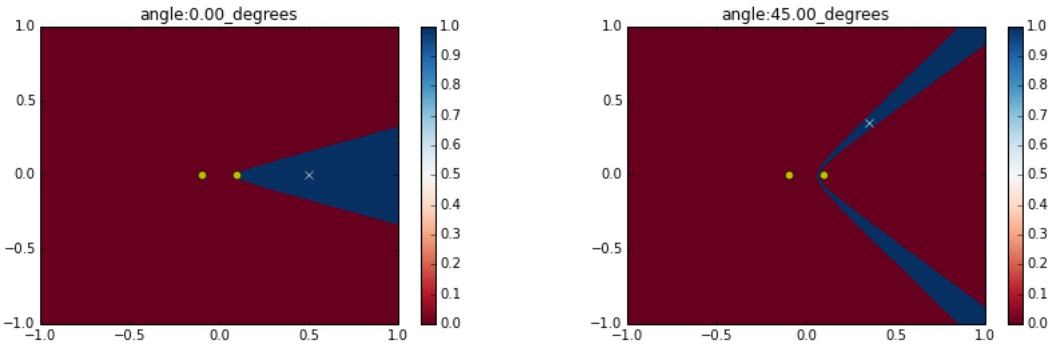
Array Architecture

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 ϵ. This ϵ 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 $2a$



(a) source at ($r = 50$ cm, $\theta = 0$ degrees) (b) source at ($r = 50$ cm, $\theta = 45$ degrees)

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:

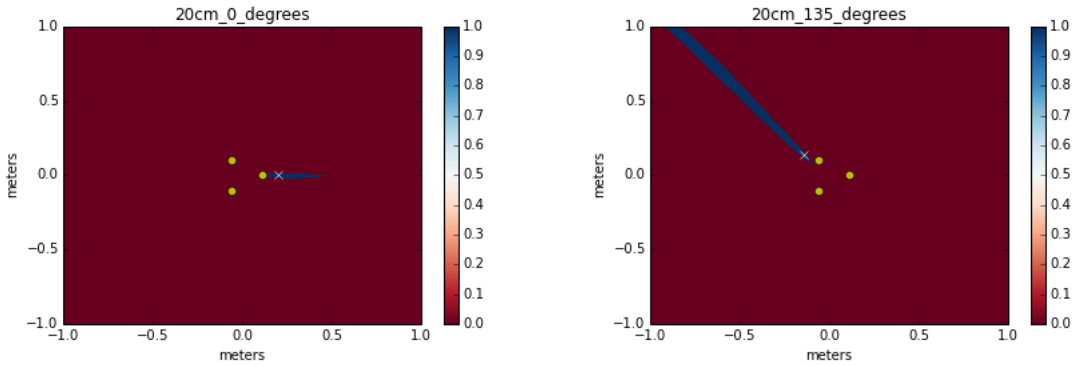
$$2a = \frac{PM_1 - PM_2}{340 \text{ m/s}}$$

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:

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

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 $M2 : (c, 0)$. All points $P : (x, y)$ with difference of distance



(a) 0 degrees (b) 135 degrees

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:

$$\frac{x^2}{a^2} - \frac{y^2}{c^2 - a^2} = 1 \tag{3.1}$$

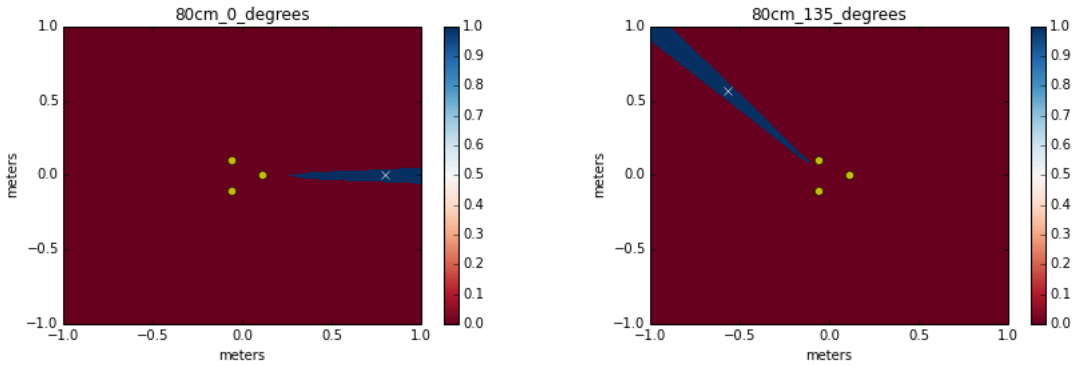
To see how the difference of distance changes with respect to source location, we can expand the equation and find the partial differential $\frac{\partial a}{\partial x}$:

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

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

$$\frac{\partial a}{\partial x} = \frac{c^2 - a^2}{\frac{c^2}{a}x - \frac{a^3}{x}} \tag{3.3}$$

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 δx would generate smaller change in difference of distance a when the source is farther away from the microphones.



(a) 0 degrees (b) 135 degrees

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

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 3000cm^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

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.

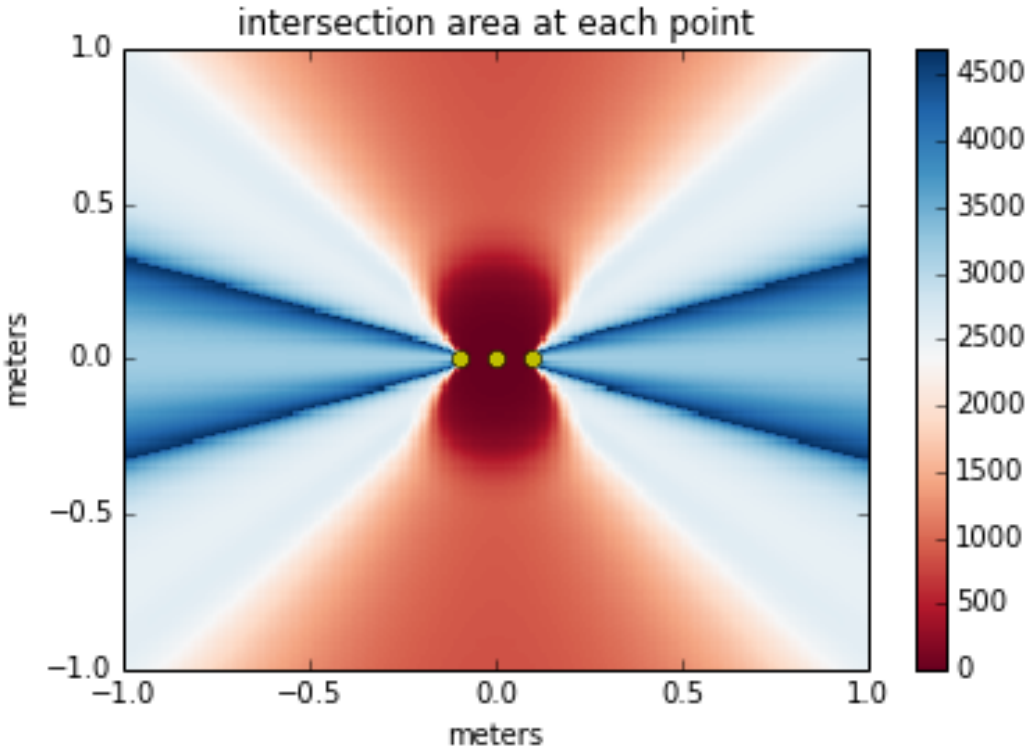
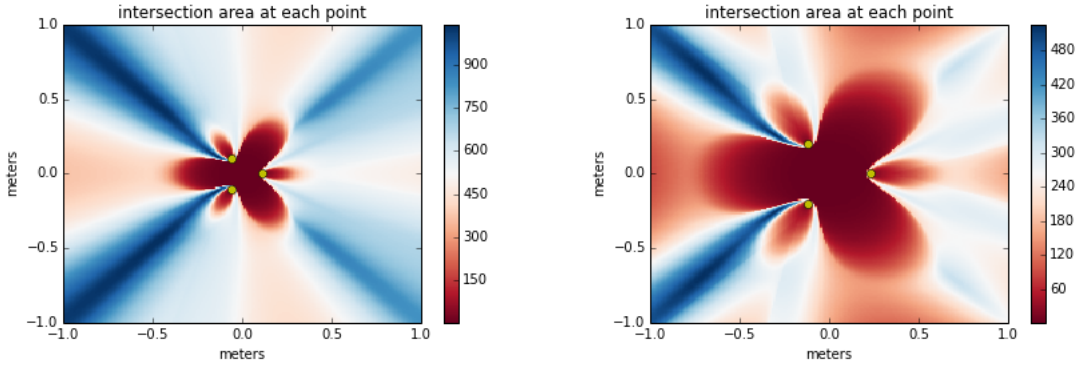


Figure 3.4: Error heatmap for different array configurations. The heatmap scale is the intersection area measured in cm^2. 3 microphones are placed in a line, 10 cm apart from each other. The average error is 55.05 cm



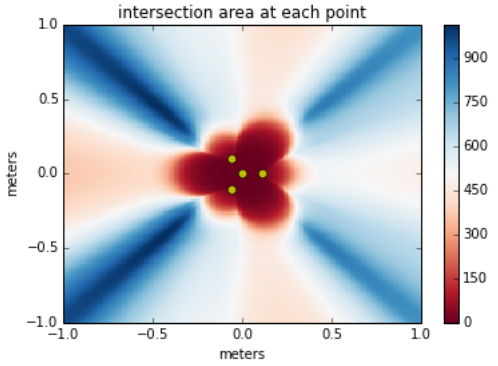
(a) 3 microphones are placed at vertices of a 20cm equilateral triangle. The average error is 18.6 cm

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

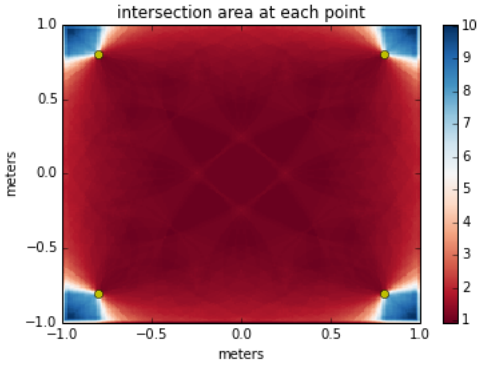
Figure 3.5: Error heatmap for different array configurations. The heatmap scale is the intersection area measured in cm^2

.

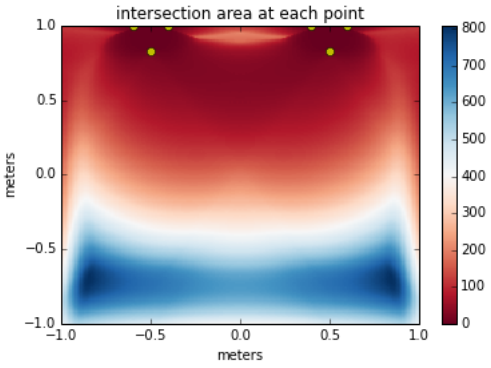
3.3 More than three



(a) Another microphone is added at origin. The average error is 17.1 cm



(b) 4 microphones are placed at 4 corners of the grid. The average error is 0.05 cm



(c) Two 3 microphone arrays are placed 1 meter apart. The average error is 2.60 cm

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

.

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 3.5a. 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 one meter by one meter region away from the two arrays. The average error is 2.60 cm. In general, synchronized clocks must be used to sample microphone data to ensure accurate comparison of arrival time differences. In practice, this means clocks must be synchronized for all the microphones across the two arrays. In the next chapter, we will describe a method that does not require clocks to be synchronized across the two arrays. This approach would make the system easier to design, although clocks still needs to be synchronized for microphones within the same array.

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, the 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

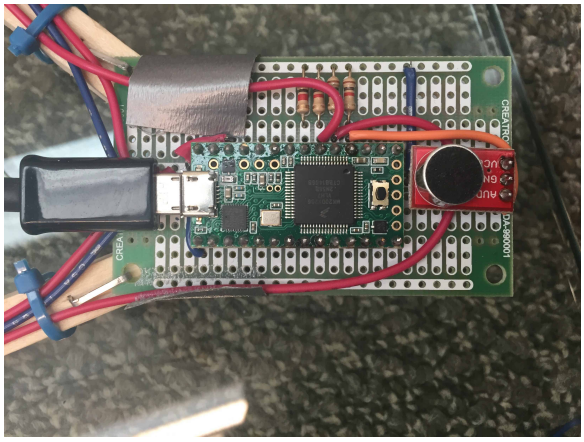
In this chapter, we first discuss the hardware equipments that are used to build the final system. Next we discuss the high level software architecture we designed and implemented in our system. We then detail the localization method we used in our system and how we circumvented the need to synchronize clocks across the two arrays. We then describe the experiments we have conducted using our system to test its localization accuracy and system responsiveness. These experiments include acoustic source location estimates and movement tracking. In the discussion section of this chapter we analyze the results from the experiments above. To conclude this chapter, we have included several interesting applications of our system demonstrating how it can be used to facilitate human computer interaction.

4.1 System

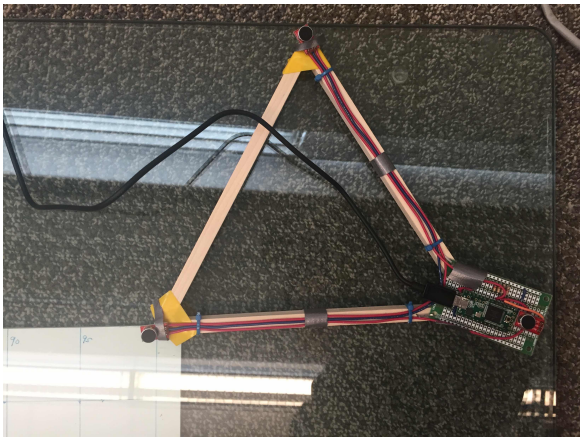
4.1.1 Hardware

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 microphone due to its low cost and small size. The microphone has a frequency range of 100 to $10K$

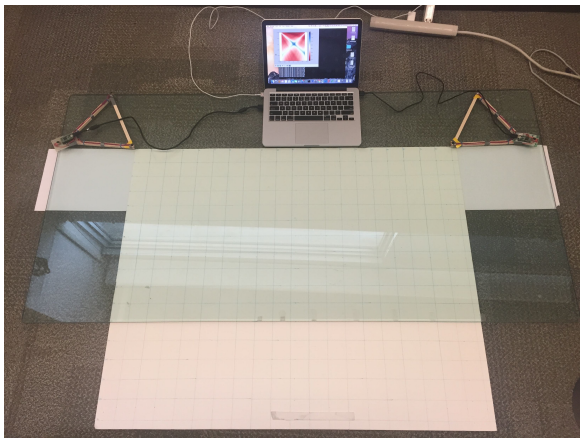
Hz, and a minimum SNR of 58dB[23]. We also used operational amplifer OPA344 from Texas Instrument to amplify the microphone output by a factor of 100, so the received signal can be easily picked up by the analog-to-digital converter (ADC) module installed on the micro-controller. The micro-controller board used in this project is *teensy 3.1*, and it is attached to one of the vertices of the triangle. Fig 4.1 shows a picture of the array setup. The micro-controller contains an onboard RAM of 64k, and an ADC module capable of sampling at 500kHz [24, 25]. In this project, the micro-controller collects microphone data on all three channels for a duration 12 milliseconds and then sends the recorded data to a computer through the USB port for localization.



(a) micro-controller



(b) array



(c) two arrays

Figure 4.1: Localization system setup

4.1.2 Software

On the software side, *Python* is used as the main programming language since it has extensive libraries in both real time data handling and signal processing. *ZeroMQ* is an inter process messaging queue that is used in our system to pass data across different modules. Our system is made up of two data acquisition modules (each is used to receive raw date from microphone array output) and one localization module (listens to both data acquisition modules and perform localization using the raw microphone data). Using ZeroMQ as connections to different modules adds flexibility to our system, as we can design applications such as the drawing application we have used in our system to interface only with the localization module and disregard how the microphone data is collected. Other localization applications can be easily integrated into our system by connecting them with the localization module. Furthermore, we have built a recording module that interfaces with the data acquisition modules for offline analysis and parameter tuning.

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. Each index i from the cross-correlation vector denotes the time delay across the two microphones receiving the acoustic signal, and the cross-correlation value k[i] at each index i denotes the likelihood of the time delay being i.

For each microphone array, we build a heatmap of likelihood for the region. The intensity at each point on the heatmap represents the likelihood of it being the source. To generate the likelihood heatmap for an microphone array, we apply the following algorithm. For each point (x,y) on the grid, the theoretical TDOA to each microphone pair can be precomputed using:

$$D_{m_1,m_2}(x,y) = \frac{((x-x_1)^2+(y-y_1)^2)^{0.5} - ((x-x_2)^2+(y-y_2)^2)^{0.5}}{v}$$

where (x_1, y_1) and (x_2, y_2) are the locations of the microphone pair and v is the speed of sound. Then the heatmap can be generated by going through all the 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:

$$\begin{aligned} L(x,y) &= R_{m_1,m_2}(D_{m_1,m_2}(x,y)) + R_{m_1,m_3}(D_{m_1,m_3}(x,y)) \\ &\quad + R_{m_2,m_3}(D_{m_2,m_3}(x,y)) \end{aligned}$$

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_1m_2, m_1m_3, and m_2m_3.

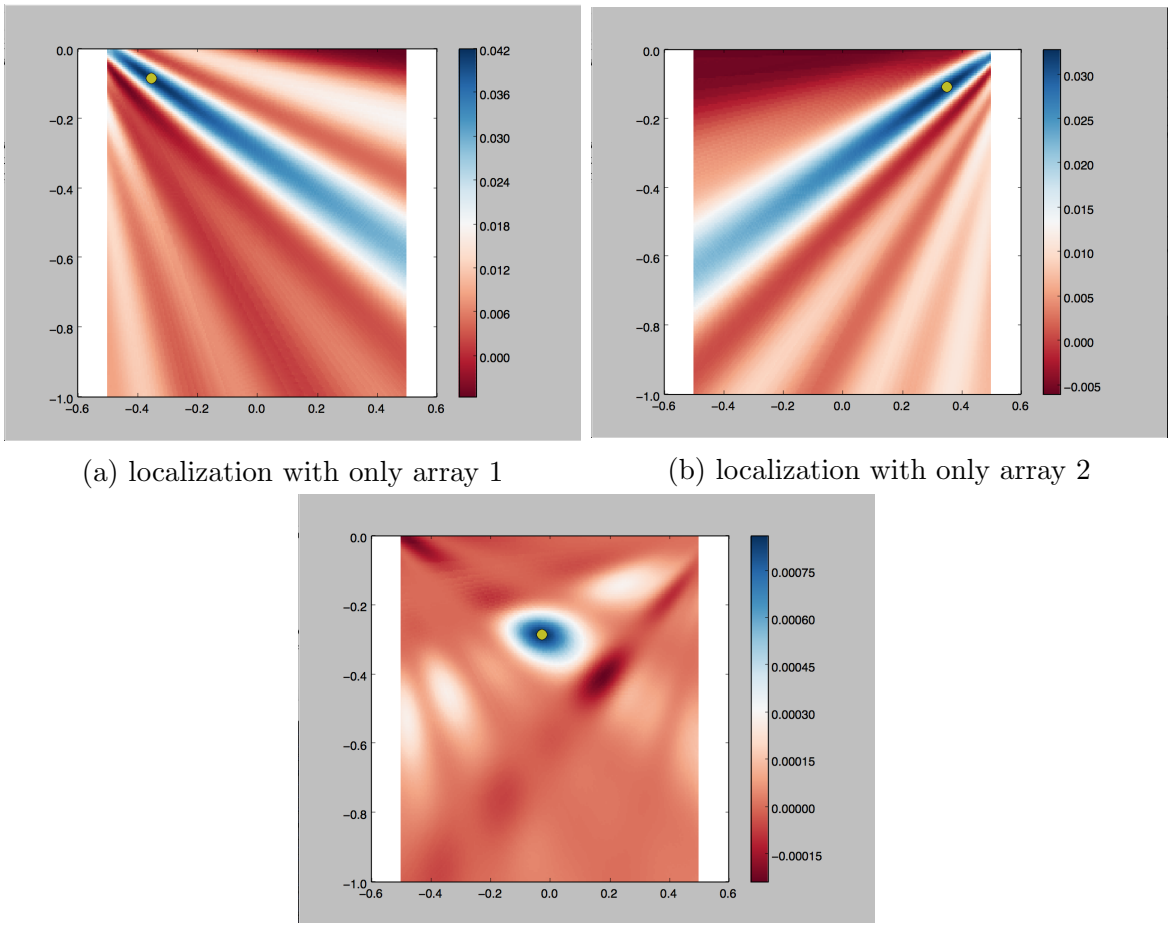
Likelihood maps from two arrays can be combined into the final likelihood map:

$$L(x, y) = L_1(x, y)L_2(x, y) \tag{4.1}$$

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). The top two figures show the individual likelihood maps produced by single microphone arrays. We can see that the individual arrays can give accurate angle estimate, but have high uncertainty in distance estimate. The bottom figure shows the combined likelihood map according to equation 4.1. 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



(a) localization with only array 1

(b) localization with only array 2

(c) localization with both arrays

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

takes around 50 milliseconds. Therefore, the total time lag between sound source and localization is around 80 milliseconds, which translates to around 12.5 localizations per second.

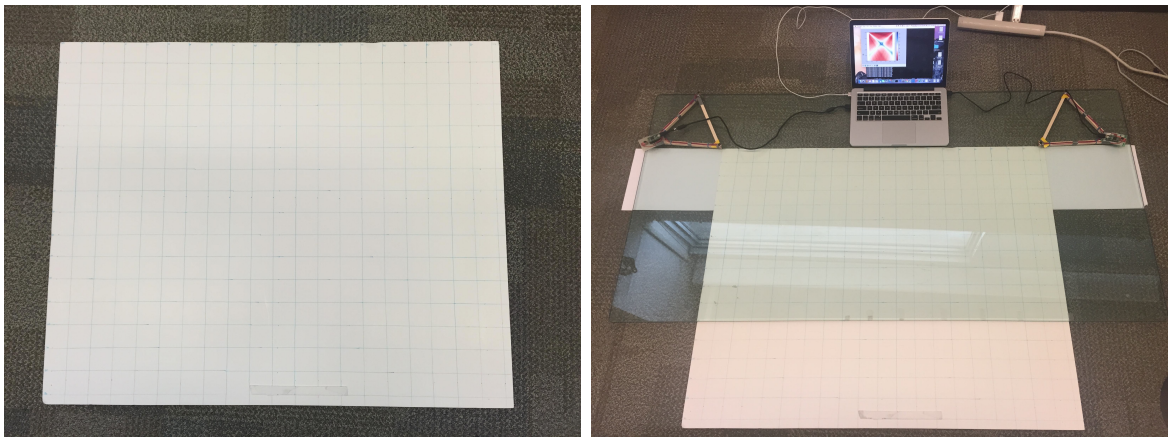
4.2 Setup

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



(a) 1 meter by 1 meter grid (b) array placement

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

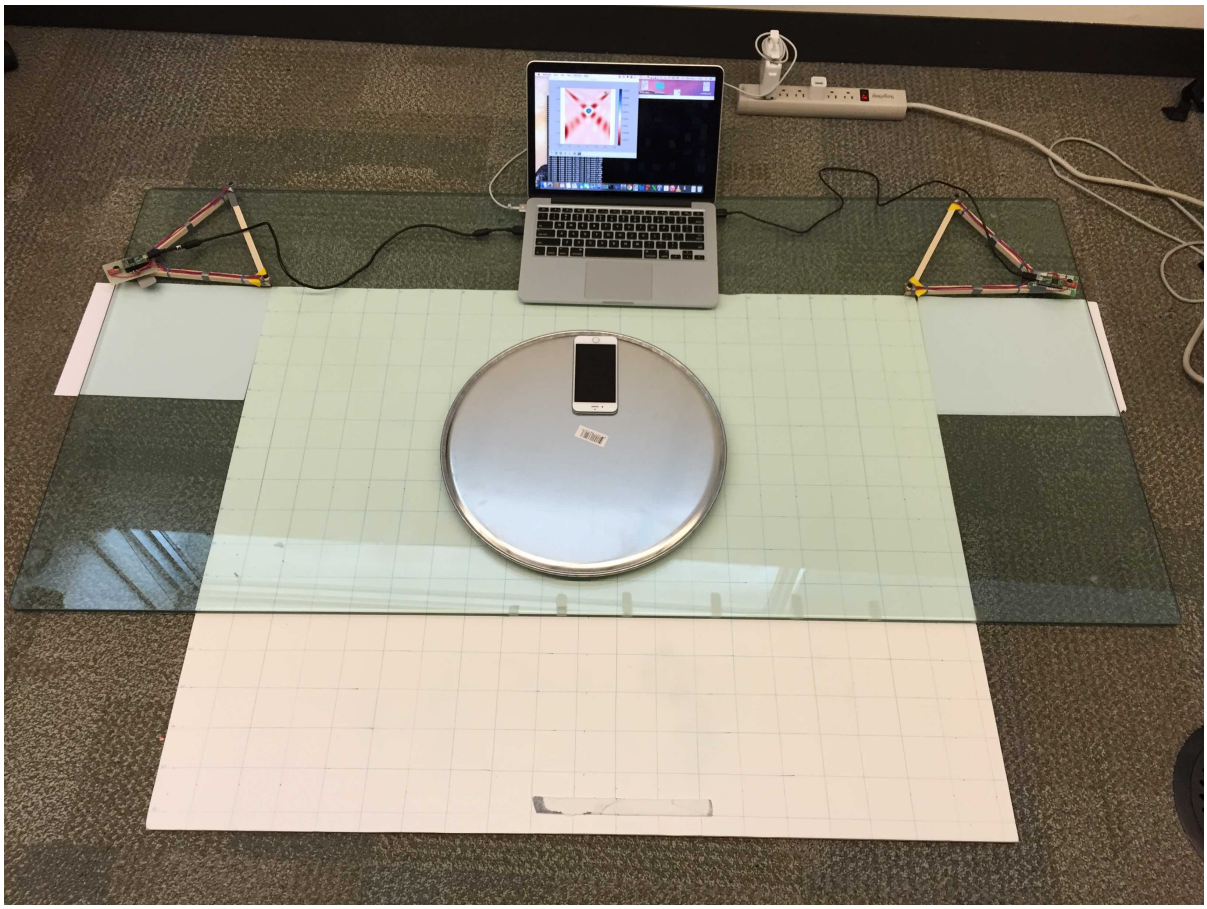


Figure 4.4: Setup for movement tracking evaluation

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_PHAT 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 know 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

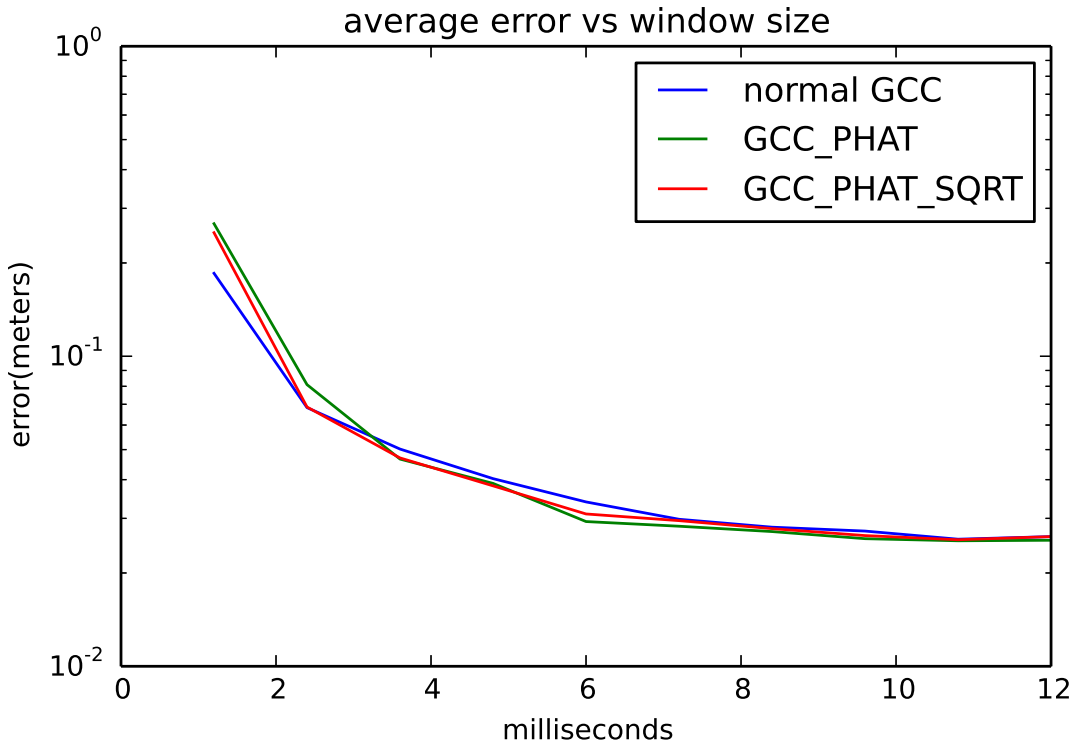


Figure 4.5: Localization error versus window size

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 when 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-

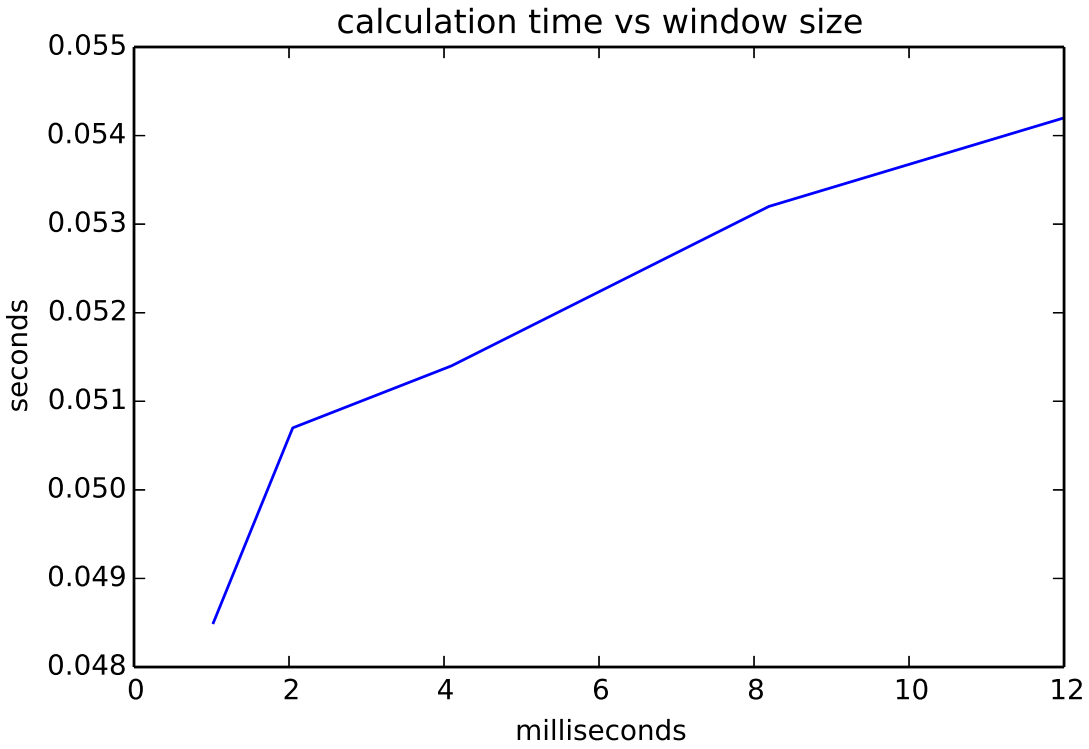


Figure 4.6: Computation time versus window size

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 cm, -10 cm) to (0 cm, -200 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.

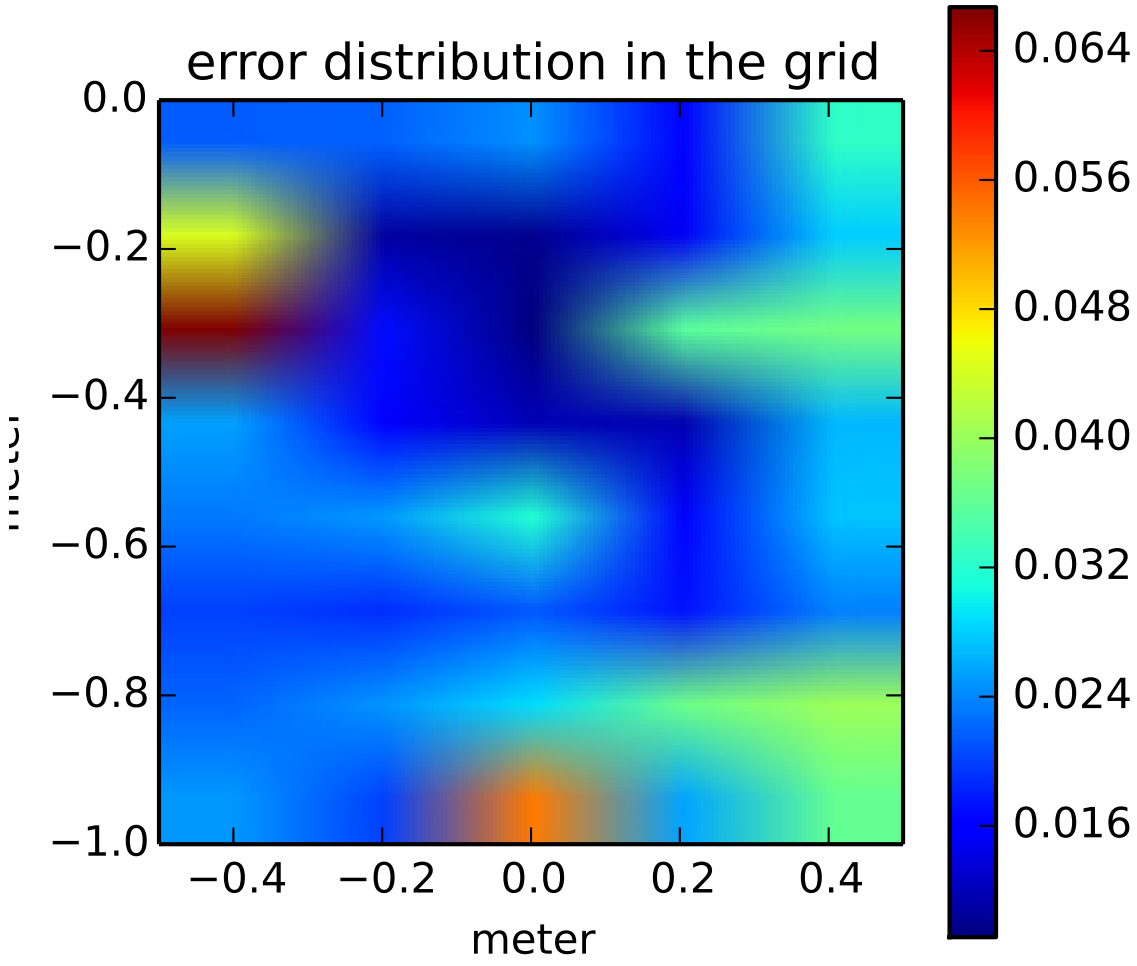


Figure 4.7: Error distribution in the grid. Arrays are placed at $(-0.5\text{ m}, 0\text{ m})$ and $(0.5\text{ m}, 0\text{ m})$

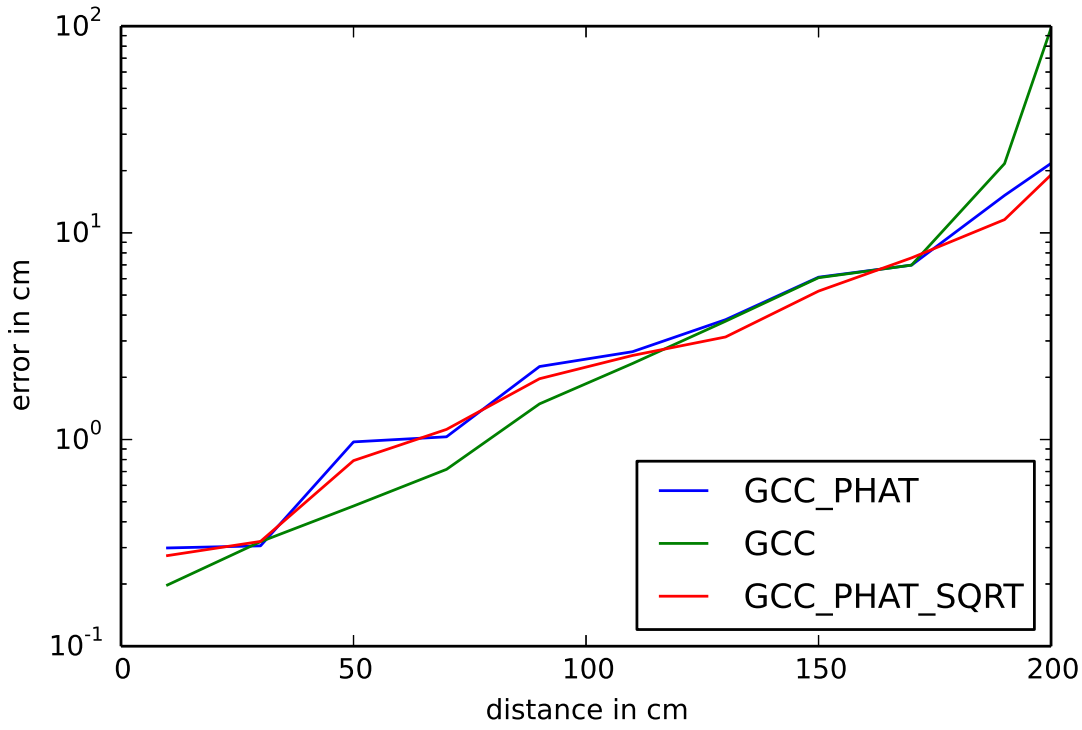


Figure 4.8: Localization error as the distance between the source and the microphone arrays increases. The source is placed on the y axis.

4.3.2 Movement tracking

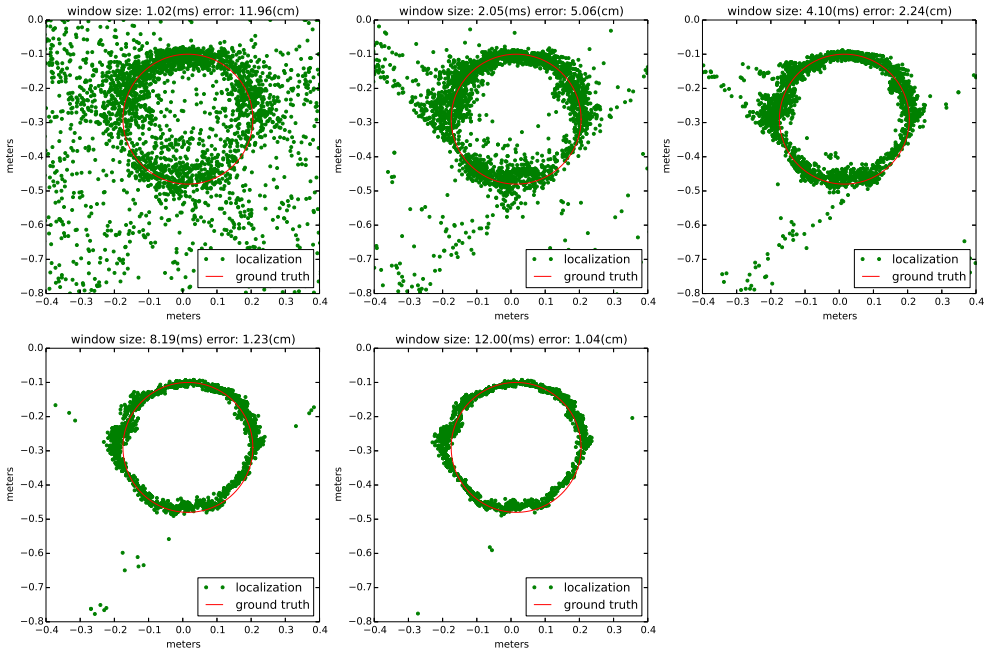


Figure 4.9: Localization quality versus window size

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

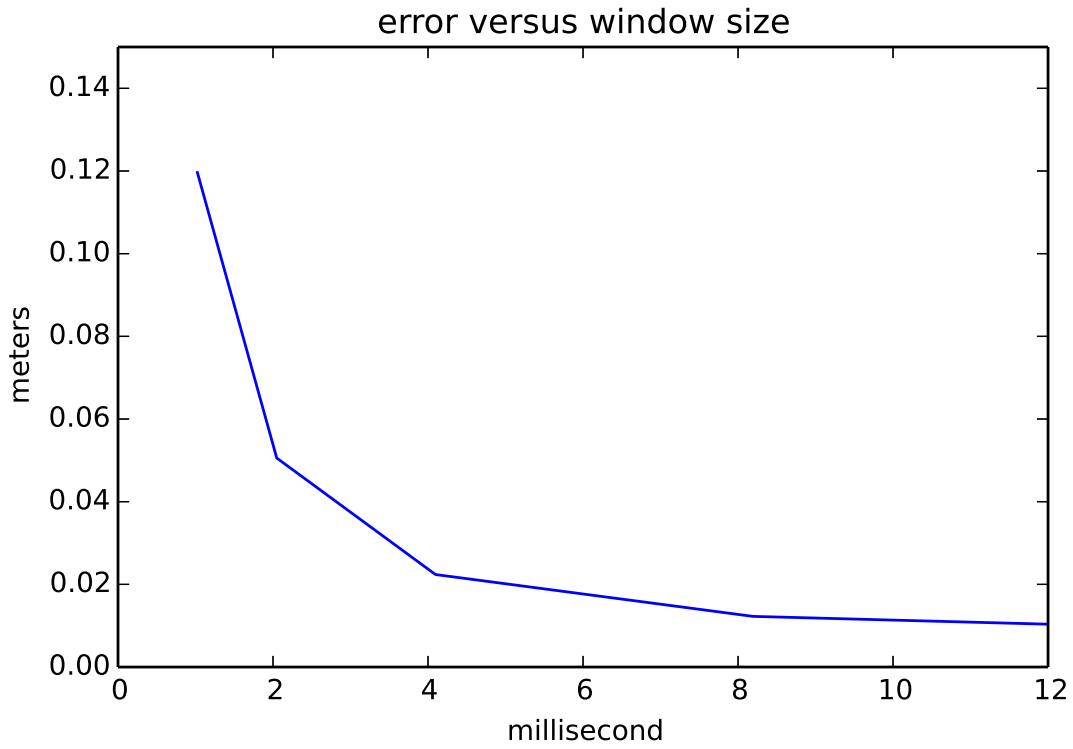


Figure 4.10: Localization error versus window size

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 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 output and averaging filtering output. 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 rotated at a faster linear speed of 20 cm per second, we find that the performance is equally good between these two movement speeds.

Fig 4.12 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 when white noise was used. The average localization error for

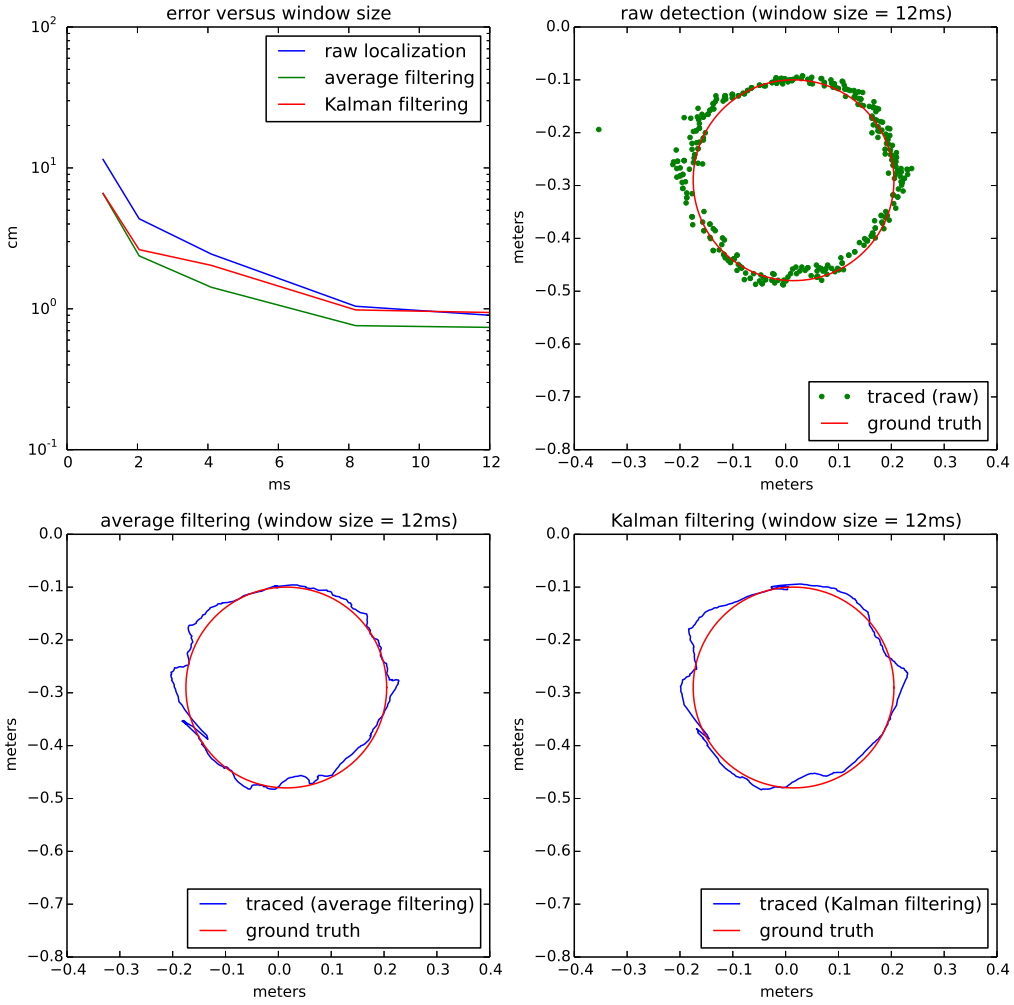


Figure 4.11: white noise (10 cm per second)

Music A is 1.289 cm. 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 4.11). Averaging filtering still produces lowest average error. Fig 4.15 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. Comparing the two experiments where music A was used as the

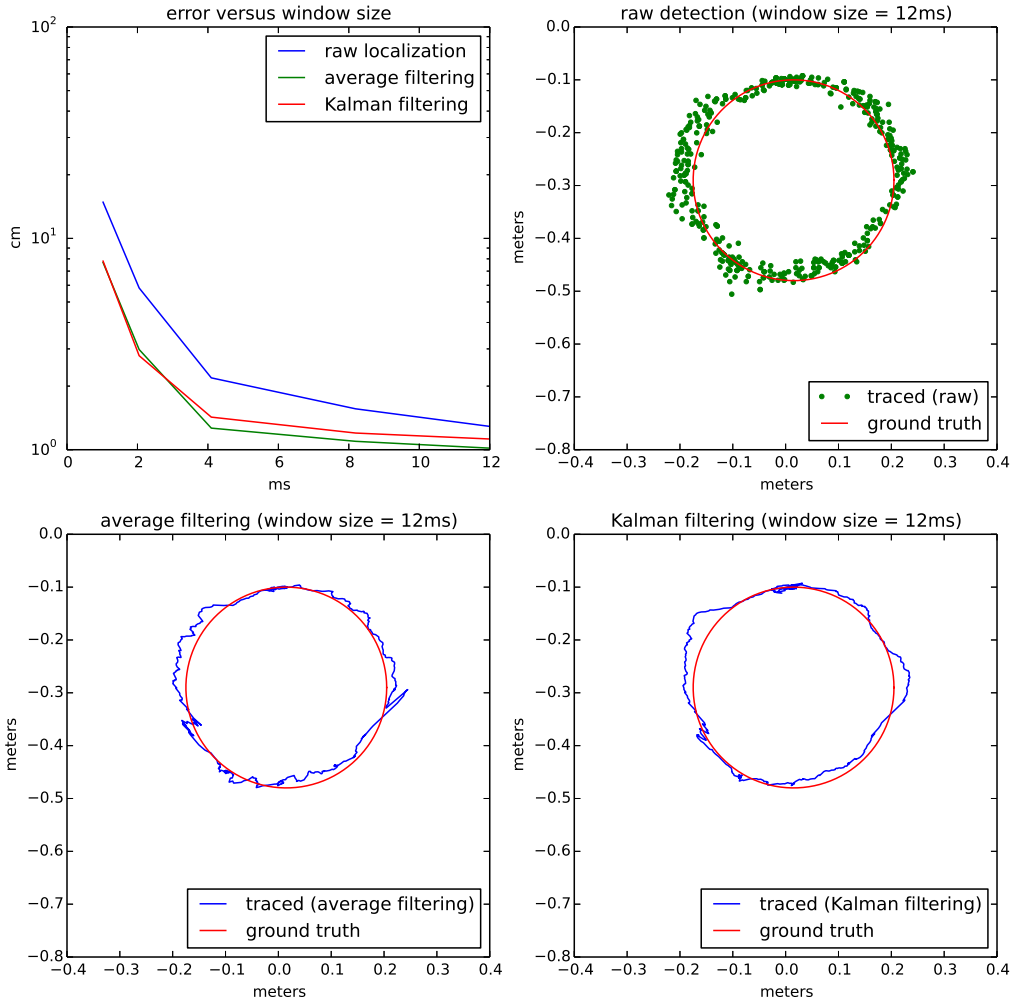


Figure 4.12: music A (10 cm per second)

sound source but with different movement speed, we find that the performance does not degrade as the movement speed increases from 10 cm per second to 20 cm per second.

Fig 4.13 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 cm. 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

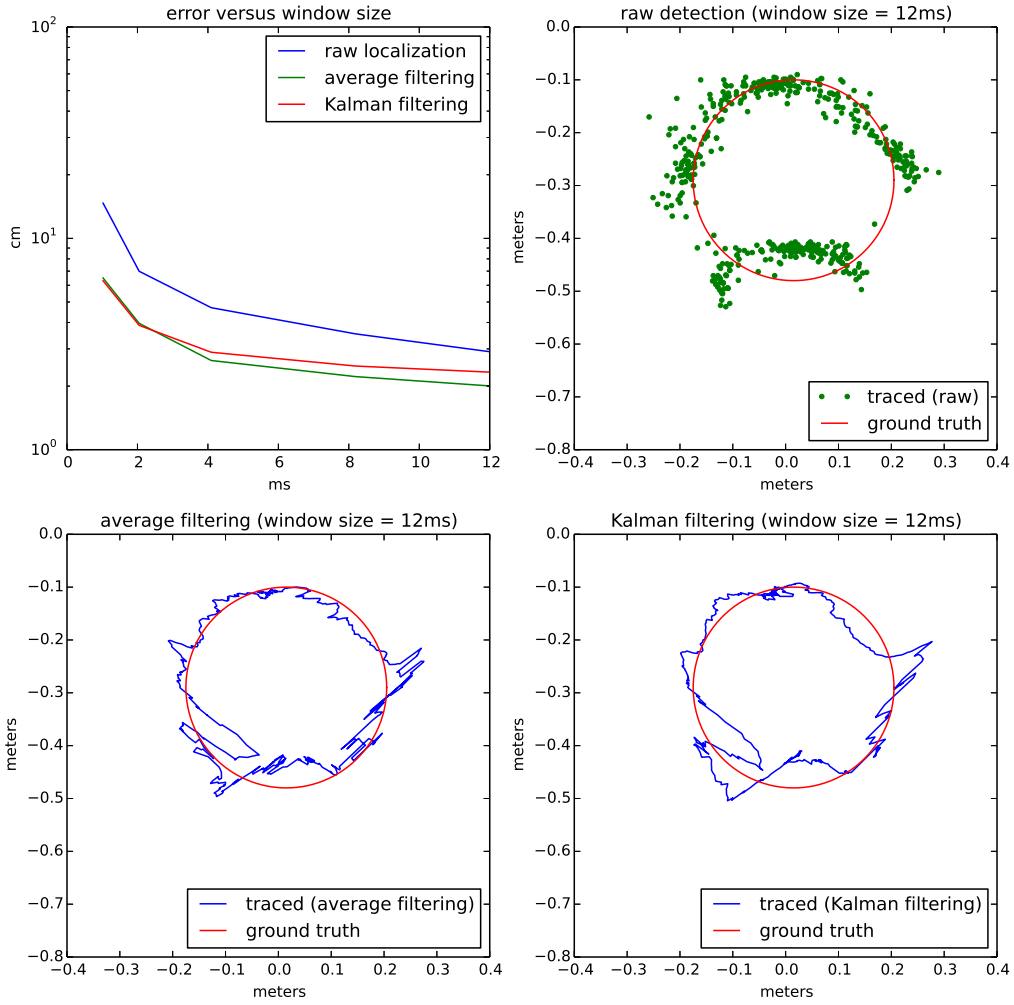


Figure 4.13: music B (10 cm per second)

raw output. The performance improvement of Kalman filtering is similar to that with Music A. Averaging filtering produces lowest average error. Comparing to the faster movement experiment of the same sound source (shown in fig 4.16), we find that the localization error is similar between these two movement speeds.

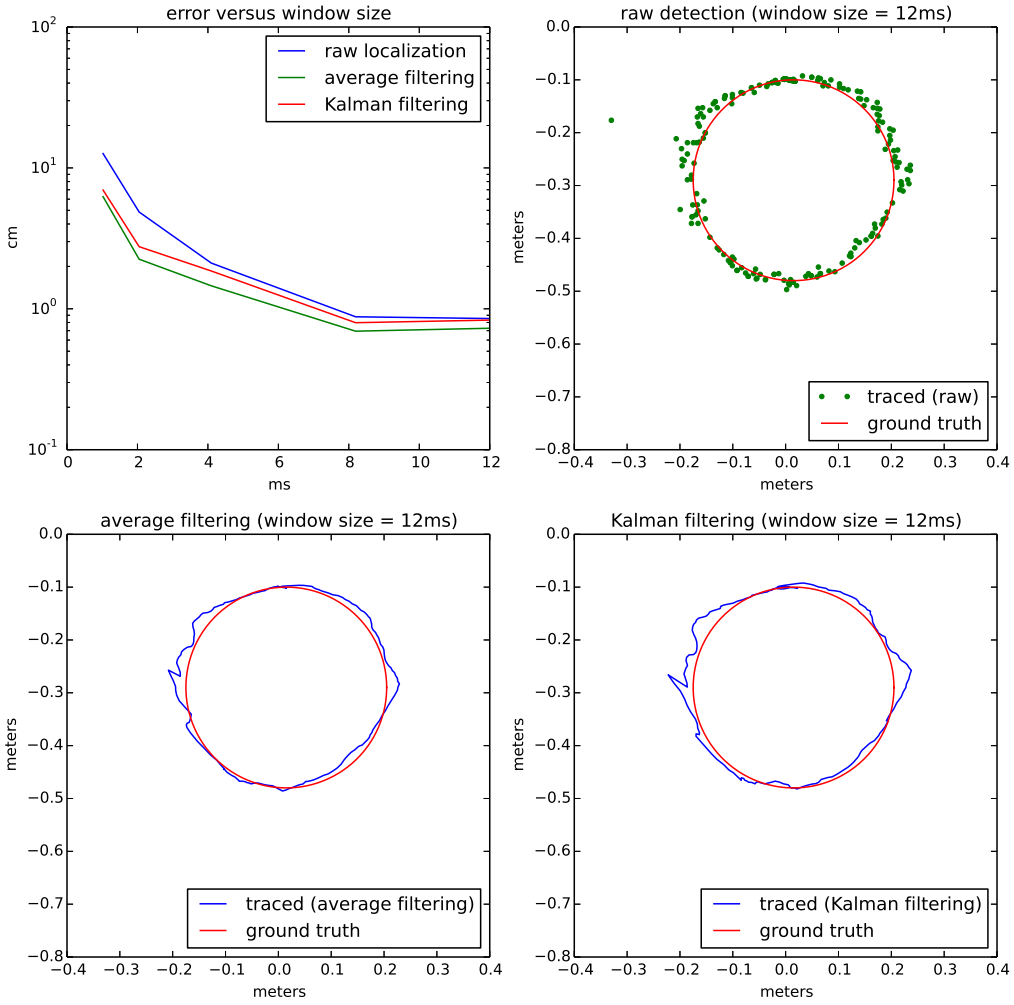


Figure 4.14: white noise (20 cm per second)

4.3.3 Discussion

Different prefiltering options produce very similar result. GCC_PHAT gives slightly better accuracy, but the difference among unfiltered GCC, GCC_PHAT and GCC_PHAT_SQRT is very small.

By comparing experiments at normal speed (fig 4.11 to 4.13) with experiments at fast speed (fig 4.14 to 4.16), we find that the localization error does not increase when the

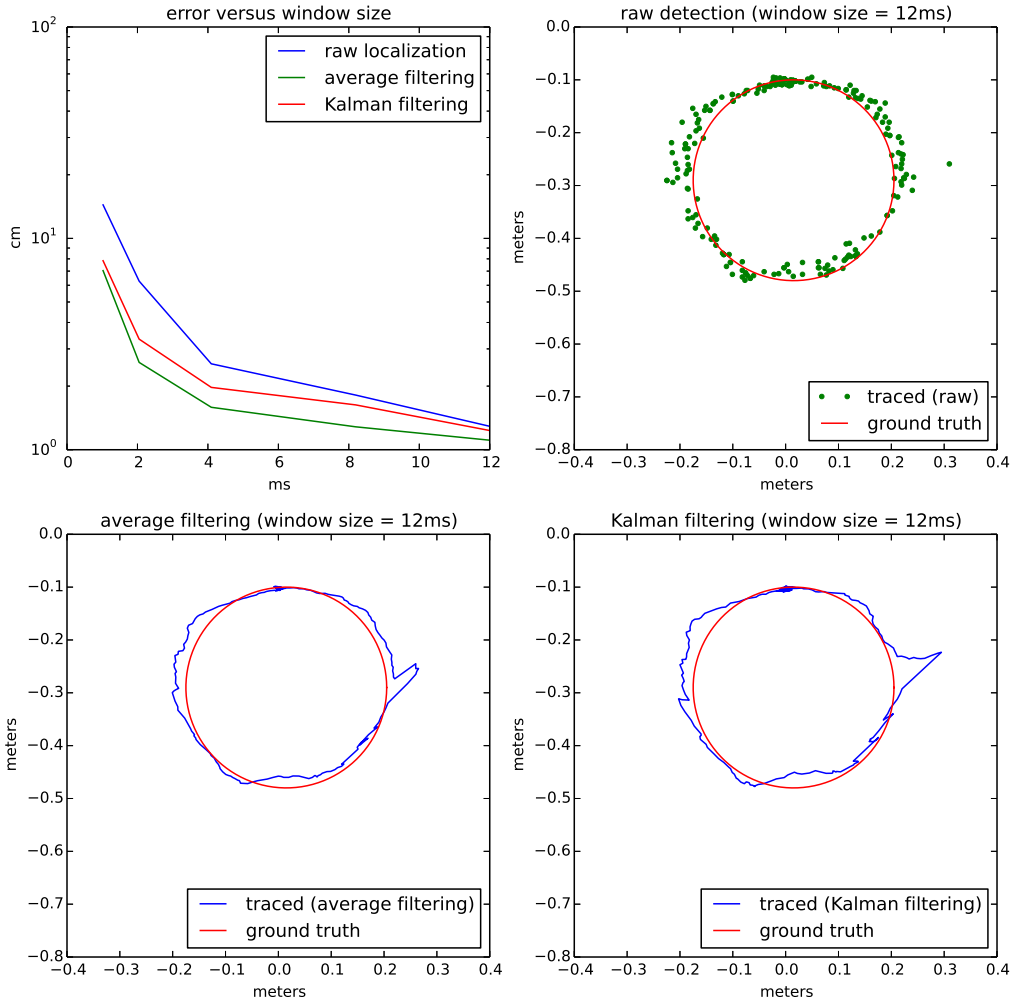


Figure 4.15: music A (20 cm per second)

movement speed increased from 10 cm per second to 20 cm per second.

When white noise is used as the audio source, Kalman filtering and raw localization produces very similar accuracy, and averaging filter gives slightly higher accuracy. However, when the audio source is changed to Music A or Music B, Kalman filtering produces better accuracy compared to raw detection, but Averaging filter still gives the best result. Looking at the smoothness of the movement path after applying different movement

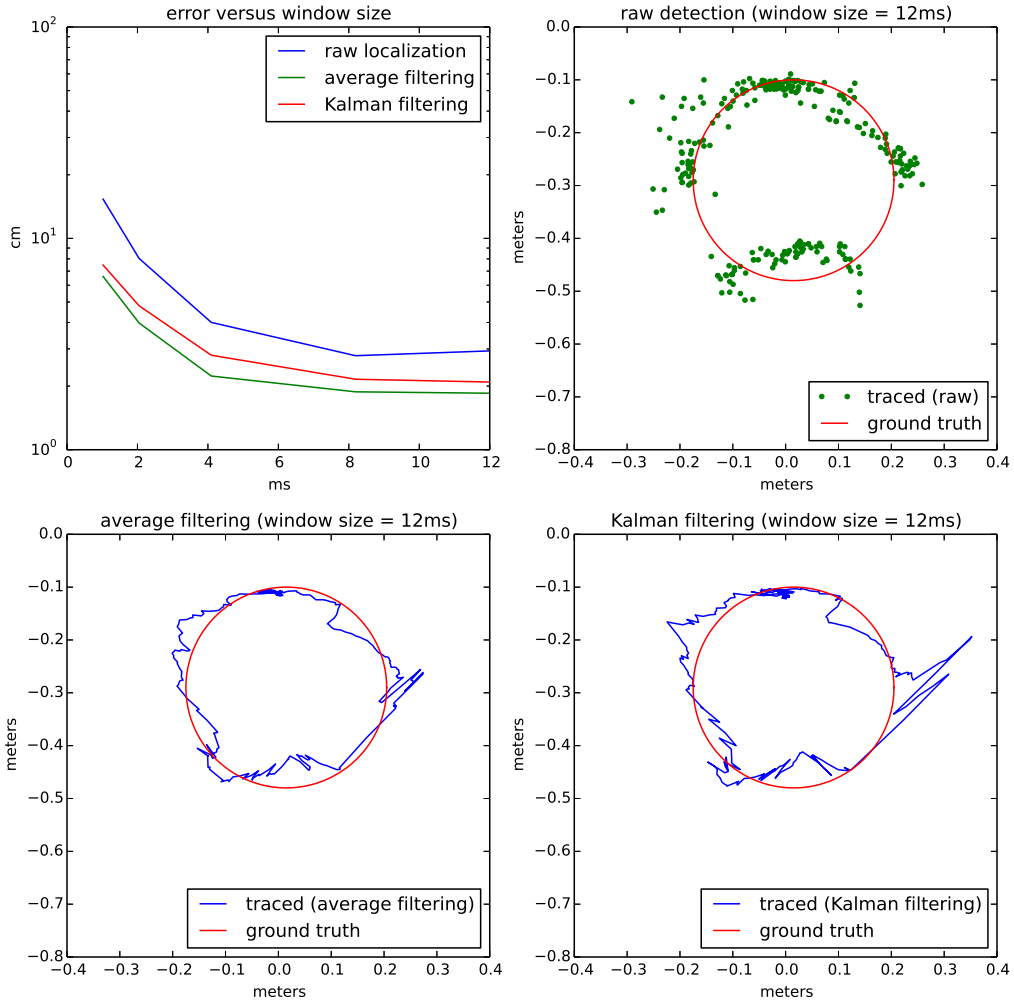
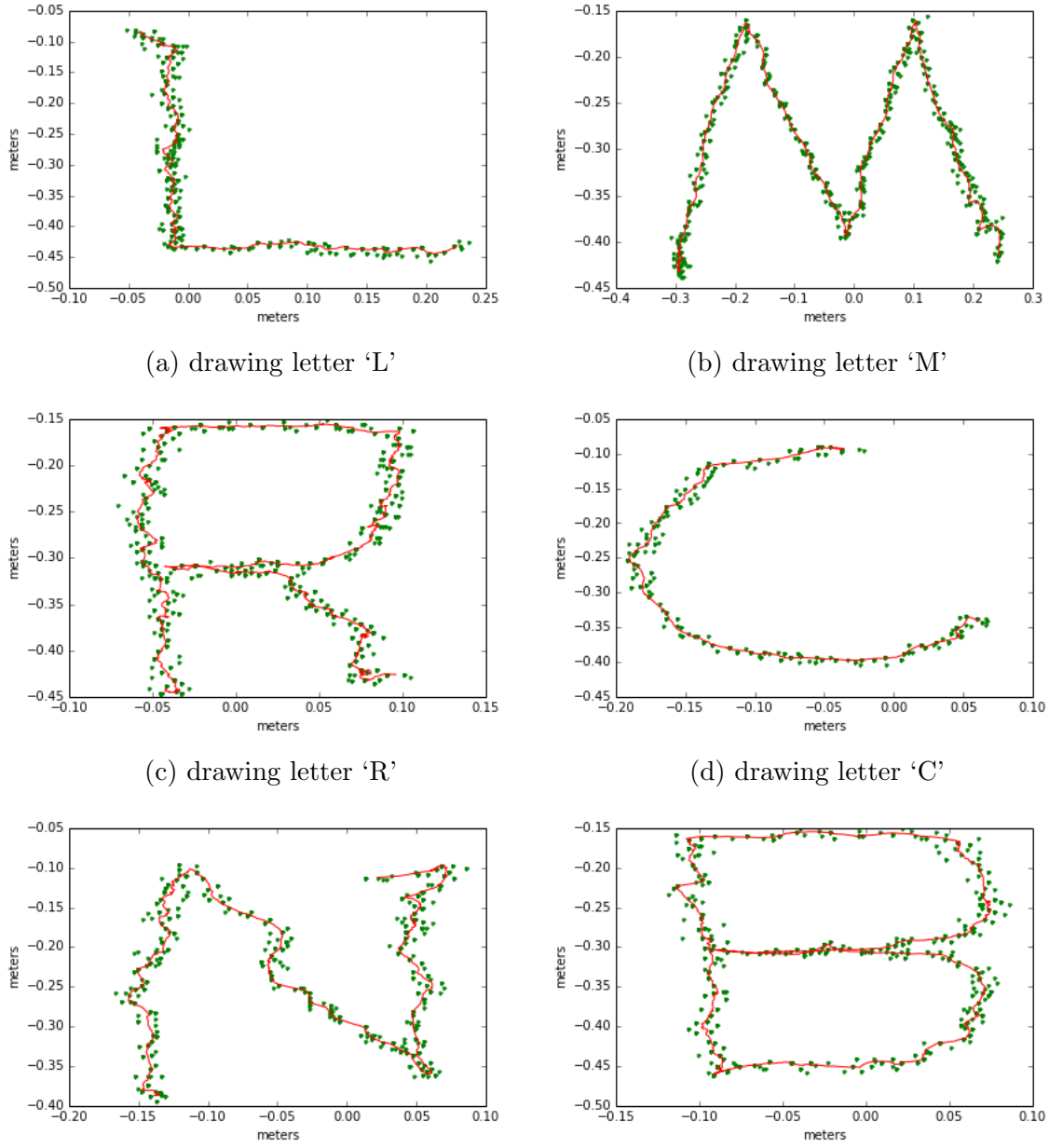


Figure 4.16: music B (20 cm per second)

filters, it can be seen 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 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.

In figure 4.17, a few examples of drawing letters with music are presented. Each letter



(a) drawing letter ‘L’

(b) drawing letter ‘M’

(c) drawing letter ‘R’

(d) drawing letter ‘C’

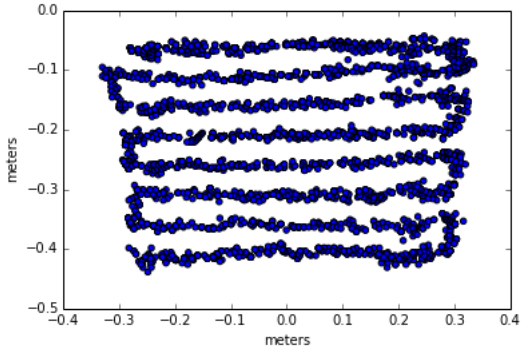
(e) drawing letter ‘N’

(f) drawing letter ‘B’

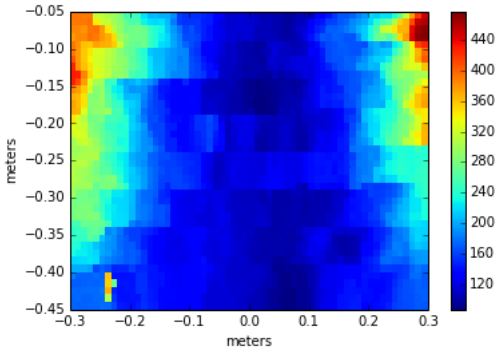
Figure 4.17: Drawing letters. Green dots represent raw localization output and the red line is the output after averaging filtering.

took around 5 to 10 seconds to draw, depending on the size of the letter and the speed of the movement. Green dots on the plots represent the raw localization output and the red line shows the movement output with averaging filtering (window size of 0.5 seconds is used). The demonstrated letters are drawn with free hand, without a track guiding the

movement. The accuracy is reasonably good, and each letter can be easily recognized. There is a bit of jiggling in the tracked movement. Part of the jiggling can be attributed to the noise from system output, and the rest comes from the hand movement.



(a) randomly traverse the region to collect location (x, y) and received power data. The volume of the source is kept constant



(b) calculate a scaling factor b_{xy} for each point in the region. The scaling factor is the average power of 5 closest points from the traversed points on the left

Figure 4.18: Volume control normalization. Two microphone arrays are placed at $(-50 \text{ cm}, 0 \text{ cm})$ and $(50 \text{ cm}, 0 \text{ cm})$.

The volume of the acoustic source can also be used to control the color of the dots painted in the above experiment. For signal $x_i[n]$ received at each microphone i, the power P_i can be used as an indicator of the source volume:

$$P_i = \frac{1}{N} \sum_{i=1}^{N} x_i[i]^2$$

Since there are 6 microphones in our system (3 in each array), we choose the maximum power P received across all 6 microphones as the indicator for the source volume:

$$P = \max_i P_i$$

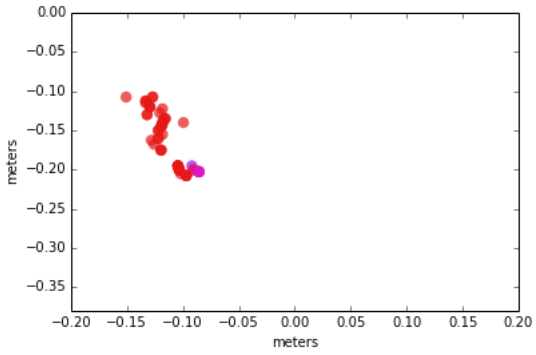
However, the received power of the audio signal is itself a function of the audio source location (since the received power decays with increasing distance between the source to receiver). Therefore, the received power needs to be normalized. For the power P

received at location (x, y), we normalize it by a scaling factor b_{xy}. The normalized power $\bar{P}$ is:

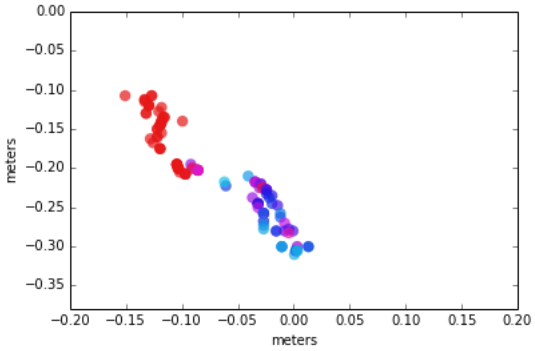
$$\bar{P} = \frac{P}{b_{xy}}$$

The scaling factor b_{xy} is location dependent. To calculate b_{xy}, we conduct an one time calibration step at the beginning. During the calibration step, a white noise sound source is used to randomly traverse the entire region with constant volume. As the source moves around in the region the calibration modules collects the source's location and the received power information. Then b_{xy} can be calculated for each location (x, y) by taking the average power of 5 closest samples from that location. Fig 4.18 shows an example of the volume control calibration procedure for calibrating the location power information in a 60 cm by 40 cm region. Fig 4.18a shows how the white noise source with constant volume is traversed in the region and fig 4.18b shows the corresponding scaling factor b_{xy} for each point in the region by taking the average power of 5 closest points in fig 4.18a. As can be seen from the figure, for any fixed y coordinate, the received power first decreases then increases along increase in the x direction. This is because the arrays are placed at the top left and top right corners of the region. For a fixed x coordinate, the received power decreases monotonically along the y direction.

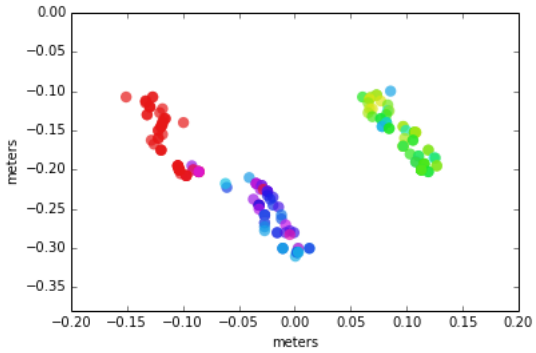
To show how the volume acts as a color control. A simple experiment is shown in Figure 4.19. In this experiment, three stripes are drawn with the same acoustic source but different volume. The color is generated in the Hue-Saturation-Value color space where the Saturation and Value are set to 0.9 while the hue is controlled by the volume. Figure 4.19a shows the first stripe where the volume is set to the maximum and the painted color is in the red to orange range. In fig 4.19b, we painted another strip after the volume is decreased by 20 percent. The painted color changed to the blue to purple range after the volume is decreased. In fig 4.19c, we painted a third strip where we further



(a) drawing a stripe with maximum volume



(b) Drawing another stripe with the volume decreased by 20%



(c) Drawing a third stripe with the volume decreased by an additional 20%

Figure 4.19: Drawing stripes where the color of each stripe is controlled by the volume of the white noise

decreased the volume by another 20 percent. In this final stripe, the color changed to the green to yellow range. A video demonstration is available at [10].

To test how sensitive the color change is to the different volumes in our system, we have conducted the following experiment. In this experiment, we moved the acoustic source clockwise in an elliptical fashion. As we are moving the source, we varied the source volume gradually and noted its color change. We started with the volume set to the maximum (the corresponding color is red). Then we gradually decreased the volume as the move to the right. After the volume is decreased to 60% of the maximum, we started to gradually increase the volume until the maximum is reached again. Figure 4.20 gives a visual presentation of the painted result. As can be seen from the figure, the top

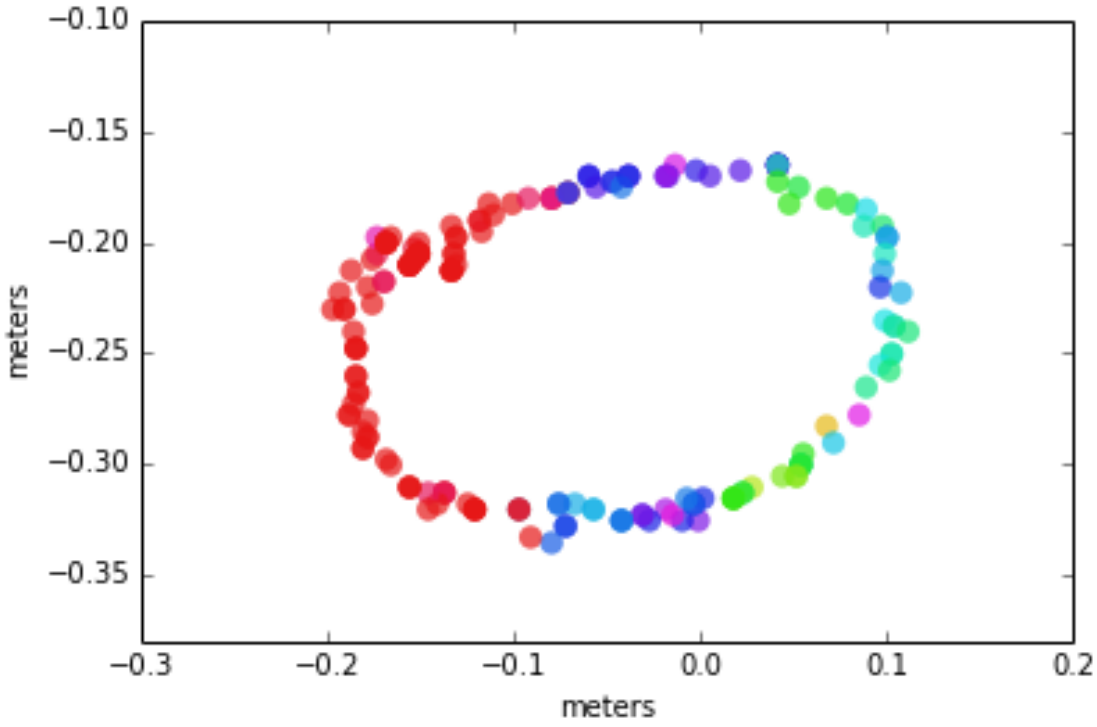


Figure 4.20: Painting an eclipse. The color of the heart changes with the volume of the song.

left region of the drawn ellipse is red. The color then gradually changes to the green yellow range and then back to the red orange range as we complete the ellipse. A video demonstration of this experiment is available at [12]

Instead of manually varying the volume of the acoustic source, a natural extension is to use music as our acoustic source. Generally, the volume of music will vary with time and this change will be reflected by the colors drawn. In figure 4.21, we used a pop song as our acoustic source and painted a heart shape by free hand. The song used here is *Don't Let It Break Your Heart* by *Coldplay*. As can be seen from the figure, the color of the heart changes at each time instant according to the volume of the song. A video demonstration is available at [11].

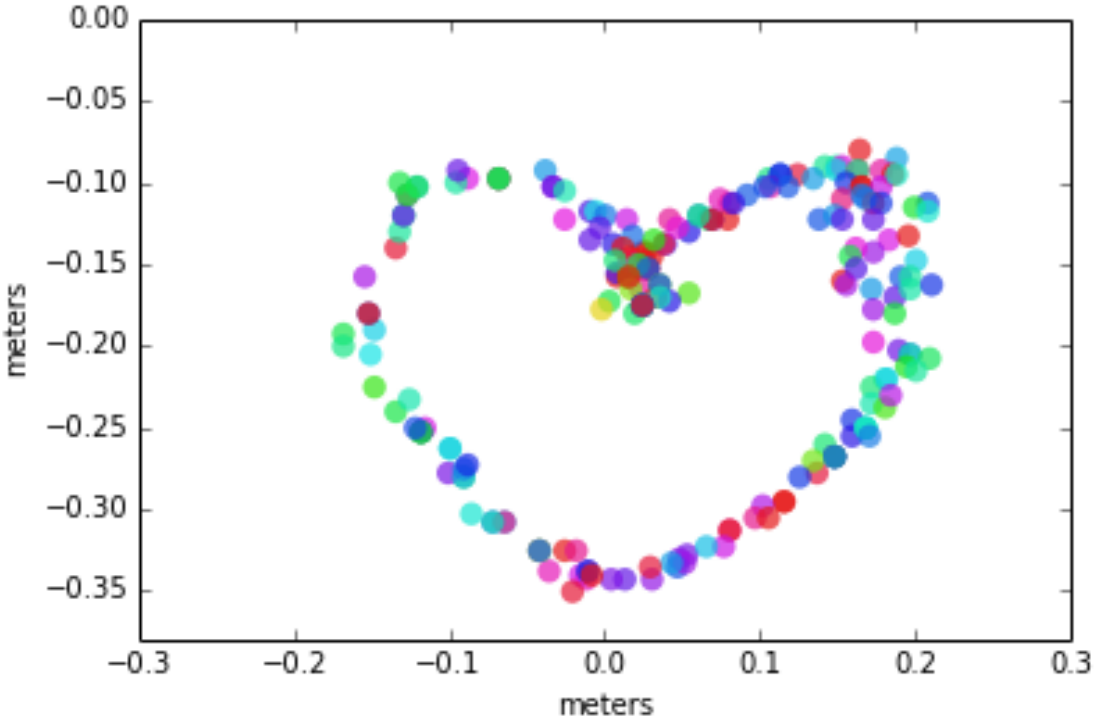


Figure 4.21: Painting a heart shape with a pop music by free hand. The color of the heart changes with the volume of the song.

Chapter 5

Conclusion

In this thesis, we described and built an inexpensive, portable yet reasonably accurate sound localization system with two microphone arrays. We have analyzed different array architectures and showed that the localization accuracy varies with the array geometry and the location of the sound source (in terms of 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 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 we have built can be installed 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 different sound sources. However, the number of frequency bands are limited in the audible spectrum, and our results showed that TDOA estimation is more accurate for a wide band signal compared to a narrow band signal.

To allow signals to occupy the entire frequency band, one potential approach is to devise protocols that allow time division multiplex of source signals, where each source signal occupies a different time slot. Another approach is for each source to have a different audio signature. If the localization module knows all sources' audio signatures, it can use that information to recover each source signal from the received superimposed waveform.

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] Video demonstration, Painting strips with white noise, URL: `https://youtu.be/khTlwEq7xbw`

[11] Video demonstration, Paiting a heart with a pop song, URL: `https://youtu.be/_4eiDYQSZEg`

[12] Video demonstration, Paiting an ellipse with white nose, URL: `https://youtu.be/RxJ-lpPqi9M`

[13] Pham, D. T., et al. "Tangible acoustic interface approaches," *Proceedings of IPROMS 2005 Virtual Conference*, 2005.

[14] 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

[15] Chowdhury, Tusi. "Single Microphone Tap Localization," *University of Toronto*, 2013.

[16] Roger R Labbe Jr, "Kalman and Bayesian Filters in Python", book, 2015.

[17] Kleeman, Lindsay. "Understanding and applying Kalman filtering," *Proceedings of the Second Workshop on Perceptive Systems, Curtin University of Technology, Perth Western Australia (25-26 January 1996)*, 1996.

[18] G. Welch and G. Bishop, "An introduction to the Kalman filter," *Dept. Comput. Sci., Univ. North Carolina, Chapel Hill, Tech. Rep. TR95041*, 2000

[19] Negenborn, Rudy. "Robot localization and Kalman filters," *Diss. Utrecht University*, 2003.

[20] 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.

[21] 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

[22] Paradiso, Joseph A., et al. "Passive acoustic sensing for tracking knocks atop large interactive displays," *Proceedings of IEEE. Vol. 1. IEEE*, 2002.

[23] Datasheet for microphone CEM-C9745JAD462P2.54R, URL: `http://cdn.sparkfun.com/datasheets/Sensors/Sound/CEM-C9745JAD462P2.54R.pdf`

[24] MicLoc, URL: `http://ruralhacker.blogspot.ca/p/micloc.html`

[25] PJRC, URL: `https://www.pjrc.com/teensy/teensy31.html`

[26] 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.

[27] Paradiso, Joseph A., et al. "Sensor systems for interactive surfaces," *IBM Systems Journal*, 2000.

[28] Checka, Nisha "A system for tracking and characterizing acoustic impacts on large interactive surfaces," *Diss. Massachusetts Institute of Technology*, 2001.

[29] 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.

[30] Knapp, C. and Carter, G.Clifford, "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1976.

[31] Aarabi, P. and Guangji Shi, "Phase-based dual-microphone robust speech enhancement," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2004.