ABSTRACT

CHUNG, SEUNG EUN. Designing Practical Mobile Interaction Interfaces through Mobile Sensing. (Under the direction of Injong Rhee and Mladen Vouk.)

Aided by the prevalence of radio-connected, sensor-rich devices such as smartphones, tablets, and wearable devices in everyday life, the realization of ubiquitous computing has become closer than ever. In ubiquitous computing, mobile devices are closely connected to each other and frequently interact with nearby devices. However, contemporary interaction interfaces are inadequate to meet the demands for frequent connectivity. Mobile devices need to interact in an unobtrusive and sensory manner. As a basic enabling technology that builds up the foundations of the new computing paradigm, this dissertation presents new interaction interfaces for mobile systems via an application-driven approach.

First, we envision a virtual trackpad interface that tracks user input on any surface near the mobile device. We adopt the acoustic signal as the medium for interaction, which can be handled by lightweight signal processing using inexpensive sensors on mobile devices. In our prototype named vTrack, the peripheral device simply emits inaudible acoustic signals through a loudspeaker, while the receiving device performs sound source localization by leveraging a multi-channel microphone array. We build a fingerprint-based localization model using various cues, such as time difference of arrival, angle of arrival, and power spectrum density of the audio signal. The vTrack system integrates the frequency difference of arrival incurred by the Doppler shift to track the sound source in motion. Finally, the position estimations are fed into the extended Kalman filter to reduce errors and smooth the output.

Further, we extend vTrack to a 3 dimensional space and propose a system named 3DTrack, which tracks a user’s free-form hand movement in 3D using the off-the-shelf mobile devices. By attaining cues from the characteristics of sound, 3DTrack’s novel localization algorithm enables mobile devices to interact with any sound-emitting peripherals including smartphones and wearable devices. More specifically, we model the 3D workspace using the time difference of arrival values observed at the microphone array. Then, we develop a candidate selection algorithm using the inverse distance law of the sound amplitude and the Doppler effect of sound frequency to accurately infer the depth of the position in a 3D space. We verify the feasibility of 3DTrack by comparing the performance with a Microsoft Kinect motion tracking sensor.
Finally, we propose a practical interaction interface for a local set of devices to connect to each other without requiring a lot of effort from their users. We find that highly time-correlated Wi-Fi signal measures experience less variation in their received signal strength (RSS), which alleviates the impediment of the RSS-based solution. We also notice that each room observes significantly discrete signal signature even when they are adjacent to each other. We leverage these observations and rely on the RSS measure to detect and unobtrusively pair devices in the same space. To achieve our goal, we devise a new similarity metric for RSS comparison and propose a means of selecting the cluster threshold. We present a prototype named Flock for Android mobile devices, which allows mobile users to interface with nearby devices in a few seconds without having to search-and-select target devices.
Designing Practical Mobile Interaction Interfaces through Mobile Sensing

by
Seung Eun Chung

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2016

APPROVED BY:

Khaled Harfoush
Mihail Sichitiu

Injong Rhee
Co-chair of Advisory Committee

Mladen Vouk
Co-chair of Advisory Committee
DEDICATION

To my family.
BIOGRAPHY

Seung Eun Chung received a bachelor's degree in Computer Science and a master's degree in Electrical Engineering and Computer Science, both from Seoul National University, Seoul, Korea, in 2007 and 2009, respectively. Since August 2010, she pursued her Ph.D. degree in Computer Science at North Carolina State University, Raleigh, NC. Her research interests include practical design and implementation of ubiquitous and wireless sensing systems for mobile interaction.
ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my advisor Dr. Injong Rhee for his thoughtful guidance and endless support in all aspects of the research process. Most of all, I value his insight for high-quality research and his patience and understanding over the past several years. I am also deeply grateful to my co-advisor Dr. Mladen Vouk and advisory committees Dr. Khaled Harfoush and Dr. Mihail Sichitiu for serving on my dissertation committee. I appreciate their generosity of time and insightful feedback.

As a member of the Network Research Lab, I would like to thank my fellow labmates for helpful discussions and comments. I thank all of the faculty and staff members in the Computer Science department for their active support towards my research. I feel fortunate to have had opportunities to work at Samsung Research America in San Jose and Samsung Electronics DMC R&D center in Suwon, Korea as an intern. I thank mentors and many other researchers at both locations. I am grateful to my graduate advisor Dr. Chong-kwon Kim at Seoul National University who inspired and prepared me to pursue this degree.

I count myself lucky to have many amazing friends here in Raleigh and also in Korea. I appreciate their constant encouragement and support over these years. Thank you all for always being there and sharing memories throughout my Ph.D. life. Finally, I cannot thank enough my family for their unconditional love and earnest support in every endeavor. Without their inspiration and encouragement amidst difficult times, my doctoral journey would not be possible. This dissertation is dedicated to them.

This work was supported in part by NSF grant # 1016216.
# TABLE OF CONTENTS

## LIST OF TABLES
- vii

## LIST OF FIGURES
- viii

## Chapter 1 Introduction
- 1
  1.1 2D and 3D Gesture Tracking for Mobile Devices  3
  1.2 Unobtrusive Interaction Interface for Multiple Co-located Devices  3

## Chapter 2 A Virtual Trackpad Interface using Sound Source Localization
- 5
  2.1 Introduction  5
  2.2 Related Work
    - 2.2.1 Sound Source Localization  8
    - 2.2.2 Mobile Input Techniques  10
  2.3 2D Sound Source Tracking
    - 2.3.1 Coordinate Positioning  11
    - 2.3.2 Movement Detection  18
  2.4 Implementation
    - 2.4.1 Audio Signal Design  22
    - 2.4.2 Signal Processing  23
    - 2.4.3 Noise Reduction  26
  2.5 Performance Evaluation
    - 2.5.1 Experiment Setup  27
    - 2.5.2 Static Sound Source Localization  28
    - 2.5.3 Moving Sound Source Localization  30
  2.6 Conclusion  33

## Chapter 3 3D Motion Tracking through Sound Source Localization
- 34
  3.1 Introduction  34
  3.2 Related Work
    - 3.2.1 Vision-based Tracking  36
    - 3.2.2 Audio-based Tracking  37
    - 3.2.3 Sensor-based Tracking  38
    - 3.2.4 RF-based Tracking  38
  3.3 3D Motion Tracking
    - 3.3.1 3D Space Modeling using TDoA  38
    - 3.3.2 Candidate Matching from DB  43
    - 3.3.3 Handling the Moving Sound Source  45
    - 3.3.4 Improving the Accuracy  46
  3.4 Implementation  48
### 3.4.1 3D Rendering .................................................. 48

### 3.5 Performance Evaluation .................................................. 50
- **3.5.1 Micro-benchmarks** ........................................... 51
- **3.5.2 Macro-benchmarks** ........................................... 56
- **3.5.3 Energy Consumption** .......................................... 58

### 3.6 Conclusion .......................................................... 59

### Chapter 4 An Unobtrusive Interaction Interface for Multiple Co-located Devices ........................................... 60
- **4.1 Introduction** ...................................................... 60
- **4.2 Related Work** ..................................................... 62
- **4.3 Coexistence Detection** ........................................... 63
  - **4.3.1 Similarity Feature Extraction** ......................... 64
  - **4.3.2 Similarity Metric Verification** ....................... 65
  - **4.3.3 Impact of Signal Fluctuation** ......................... 69
  - **4.3.4 Impact of the Number of APs** ....................... 71
  - **4.3.5 Co-located Device Discovery** ....................... 72
- **4.4 System Implementation** .......................................... 74
  - **4.4.1 In-phone Processing** ................................... 75
  - **4.4.2 In-cloud Processing** ................................... 76
- **4.5 Evaluation** ...................................................... 77
  - **4.5.1 Discovery Performance** .................................. 77
- **4.6 Conclusion** ...................................................... 79

### Chapter 5 Conclusion ...................................................... 80

### BIBLIOGRAPHY .............................................................. 81
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Anthropometric dimensional data of shoulder-to-wrist length in cm.</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Tracking results according to different tracking time show 3DTrack's position estimation error does not accumulate over time.</td>
<td>56</td>
</tr>
<tr>
<td>3.3</td>
<td>Energy consumption measured during one-hour application run. Case 1 keeps only the screen on. Case 2 turns both screen and audio recording function on. Finally, case 3 activates all functionalities including coordinate computation, audio signal processing and filtering.</td>
<td>58</td>
</tr>
<tr>
<td>4.1</td>
<td>Correlation coefficients between the ground truth and multiple dataset using different distance metrics.</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Time to get a Wi-Fi scan result and the corresponding peer discovery time for different test devices.</td>
<td>78</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1  vTrack operation scenario using two mobile devices. Receiver records the audio signal using three on-device microphones and visualizes the sound source position by a cursor on its screen. Transmitter emits inaudible (19kHz) acoustic signals repeatedly through an external speaker. User moves the sound source on a virtual trackpad grid. ................................. 7

Figure 2.2  Normalized time difference of arrival (top) and power ratio values (middle and bottom) of audio signals recorded by two microphones. Measurements are taken at each point of the grid. .................. 12

Figure 2.3  Distribution of cross-correlation index value and power ratio of audio signals measured on two different days (solid lines: first day, dotted lines: second day). .............................................. 13

Figure 2.4  Angle of arrival (θ), range (r), and perpendicular distance (y) estimation using the time difference of arrival (Δt) measure at two microphones. .......................... 15

Figure 2.5  Coordinate estimation of the sound source S(Xs, Ys) using the distance to three microphones M1, M2, and M3 presented on a Cartesian coordinate system. ......................... 16

Figure 2.6  Change in frequency (top) and amplitude (middle) of the audio signal observed by two microphones during the sound source movement. Velocity estimation (bottom) of the moving sound source using the Doppler shift. .......................... 18

Figure 2.7  vTrack's operation flowchart on the receiving mobile device. Audio signal processing module performs frequency-domain analysis on the audio signal. Coordinate computation module obtains various cues from the signal for coordinate positioning. ................................. 23

Figure 2.8  Sample waveform of 2ms-long sine signal emitted every 21ms by the transmitting device. .......................... 24

Figure 2.9  Ambient noise measured at a local Starbucks cafe. Various machine sounds result in a high-frequency (~19kHz) background noise. ................. 26

Figure 2.10  vTrack's noise reduction procedure builds a noise model by recording the background noise and performs spectral subtraction on the original audio signal through frequency-domain analysis. ............... 27

Figure 2.11  (a) Angle of arrival and (b) range of the sound measurements with various angle and distance parameter combinations. All error bars stand for standard deviation. ...................... 28

Figure 2.12  Measurement points and results presented using the angle and range values on a polar coordinate system. Gray zone represents vTrack's trackpad area. .......................... 29
Figure 2.13 Distribution of position estimation error in (a) static and (b) moving sound source scenarios. .................................................. 30
Figure 2.14 Tracking results composed of straight line and curve movements. .... 31
Figure 2.15 Distribution of position estimation error with a moving sound source in different background noise scenarios. ......................... 32
Figure 2.16 Two additional input applications using vTrack system: (a) a larger virtual trackpad workspace with 20×26cm grid size, (b) an Apple keyboard layout printed on 18×8cm paper. .......................... 32

Figure 3.1 3DTrack’s application scenario for a virtual reality (VR) device setting. A head-mounted mobile device tracks the movement of sound-emitting wearable devices such as smart watches. ............................ 35
Figure 3.2 Experimental setup of 3DTrack using a 3D cuboid testbed. Receiving device is attached in front of the cuboid workspace, while the sound source is allowed to move inside of the cuboid. .................. 39
Figure 3.3 3D coordinate of a sound source is defined by altitude, azimuth angles, and distance from the receiving mobile device. .................... 40
Figure 3.4 Distance differentials between theoretical distance and actual measurement depending on distance, azimuth, and altitude parameters. All error bars indicate standard errors. ......................... 41
Figure 3.5 Modeling results of 3D workspace database using theoretical time difference of arrival values computed by different pair of microphones. 42
Figure 3.6 Time difference of arrival measures by two microphones 1 and 2 can localize a sound source on a surface (hyperboloid 1) in 3D space. With three microphones, the position is identified on a line (shown in red), which is the intersection of two hyperboloids 1 and 2. ............ 43
Figure 3.7 Index estimation result of k-nearest neighbor matching when using time difference of arrival measures applied to the 3D database in a static sound source scenario. ............................ 44
Figure 3.8 Candidate selection algorithm of moving sound source scenario. Based on the inverse distance law of sound, change in sound amplitude is used to infer the distance of sound source from the receiver. ....... 45
Figure 3.9 Azimuth and altitude angle of arrival computation in 3D using the time difference of arrival (Δt) measure at two microphones. ......... 47
Figure 3.10 Distribution of azimuth and altitude angle estimation error in 3D. .... 47
Figure 3.11 3D perspective projection for 3D rendering on a 2D screen. Sound source P in 3D workspace is projected onto 2D surface S (S’) from the observer’s eye position E in 3D. ........................ 49
Figure 3.12 Tracking results showing movements with minute differences in direction and altitude in 3D space. .......................... 50
Figure 3.13  Distribution of position estimation error for 3D tracking. 3D measure includes three-axes performance, while 2D measure computes accuracy for only two axes. .......................... 51
Figure 3.14  Raw tracking data and calibrated tracking result by applying extended Kalman filter. ................................................................. 52
Figure 3.15  Tracking results performed on a virtual surface in three-dimensional space and their projection results on a two-dimensional plane. ........ 52
Figure 3.16  Distribution of position estimation error with different x, y, and z-axis parameters. ................................................................. 53
Figure 3.17  Distribution of depth estimation error compared to the ground truth depth change during the movement. ................................. 54
Figure 3.18  Distribution of distance estimation error compared to the ground truth distance traveled during the movement. .......................... 55
Figure 3.19  3DTrack’s tracking results compared with Kinect motion sensor. .......... 57
Figure 3.20  Distribution of position estimation difference between 3DTrack and Kinect motion sensor. .......................................................... 57

Figure 4.1  Experiment setup in an engineering building. 116 measurement points (corridor: labels 1~56, classroom 1: labels 57~72, classroom 2: labels 73~88, classroom 3: labels 89~116) are spaced 2.4 meters apart. Each color maps a color-labeled AP and its signal coverage, while black-labeled APs are targeted to cover each classroom they are located. .... 65
Figure 4.2  Proposed similarity metric and the Spearman rank-order correlation computed between point 1 and all other points in the corridor. .... 66
Figure 4.3  Heatmaps of (a) the physical distance and (b) dissimilarity measures between all measurement points in the corridor. ......................... 67
Figure 4.4  Heatmap of similarity measure for measurement points in the corridor (labels 31~56) and the adjacent classrooms (labels 57~116). ............ 68
Figure 4.5  Distribution of moving standard deviation of RSS for three different window length. ................................................................. 69
Figure 4.6  Distribution of similarity measure between two Wi-Fi scans performed within various window length. ......................................... 70
Figure 4.7  Distribution of similarity measure between the reference RSS reading and different number of cumulative scan results. ................. 71
Figure 4.8  Impact of the number of overlapping APs and their RSS in dBm to the similarity index. ................................................................. 72
Figure 4.9  Hierarchical clustering result using the proposed threshold strategy in different locations (labels 1~2: classrooms, labels 3~6: stores, labels 7~10: rooms in an apartment building). ........................................... 73
Figure 4.10  Server collects Wi-Fi signature from clients and clusters them by computing similarity of signatures. Then, the server connects clients in the cluster using messaging protocol. Clients perform Wi-Fi scan and communicate over the network. .......................... 75

Figure 4.11  Number of devices discovered in the same group and success ratio of experiments performed in various scenarios. .......................... 77
Chapter 1

Introduction

Mobile devices are a pervasive part of our everyday life. In ubiquitous computing, mobile devices are closely connected to each other and frequently interact with nearby devices. These pervasive devices affect the way we interact, share, and communicate with other people as well as other devices.

Mobile interaction studies these new aspects of interaction on mobile devices, which have many factors that differentiates them from traditional personal computing by their small sizes, ubiquitous uses, and frequent connectivity. In this dissertation, we focus on two representative aspects of mobile interaction: control mobile devices through their input modalities, and share data and communicate with other devices.

We pinpoint limitations regarding these two factors. First, the small-size input interface of mobile devices limits the user experience in controlling them. For example, software keyboards are error-prone due to their small sizes compared to the size of one's fingers. Also, fingers themselves block the touchscreen during the interaction. Second, mobile interaction demands a sensory and non-intrusive connectivity, because current methods involve time-consuming steps for searching and selecting the target device in the scan results. Also, dedicated and compatible hardware equipped on both devices are required to establish connection, where delicate physical conditions need to be met. For example, NFC provides 10 centimeters of communication range, while Bluetooth supports 30 centimeters to establish one-to-one connection between two nearby devices.

There have been several near-device interaction technologies proposed to tackle these inconveniences. Many off-the-screen mobile input methods are already available in the market, showing the high demand for new mobile interaction interfaces. For example, there
are some Bluetooth keyboards that allow typing input, while smartpens let users take notes on any surfaces near the device. Regarding the connectivity between devices, many applications in the app store establish connection using various cues, which recognize simultaneous gesture events to share the data. For example, bumping or shaking mobile devices at the same time allows users to exchange their name cards or pictures.

We pay attention to the advancements in mobile device hardware especially the sensors. Modern audio sensors allow high definition sensing on mobile devices. Stereo recording is available by leveraging multi-channel microphones. Active noise cancellation with dedicated microphone is supported to enhance the quality of phone calls. Also, most mobile devices use 24-bit audio data with 192 kHz audio sampling rate, which is 4 times of performance enhancement that is recently made in terms of the audio sampling rate. Additionally, a high-quality loudspeaker produces flawlessly clean audio output. Wireless sensing module provides fast and low-power dualband sensing. We find that many of the challenges in mobile interaction can be overcome by adopting sensing capability of mobile devices.

This dissertation aims to realize mobile interactions through mobile sensing on off-the-shelf mobile device by using the minimum set of on-device sensors. We achieve our goal via light-weight real-time processing, in terms of both computation and power consumption of device. Obviously, the proposed methods accomplish high accuracy in delivering user’s intention for interaction. We propose three practical mobile interaction interfaces. First, we present near-device two dimensional sound tracking by sound source localization technique. Second, we extend it to three dimensional space and achieve 3D motion tracking using acoustic signals. Finally, proximity detection technique for multiple co-located devices is proposed.

This dissertation makes contributions to several aspects in designing practical and intuitive interaction interfaces for mobile systems. First, we exploit and demonstrate mobile sensing capabilities in the context of mobile interaction using mobile devices. We adopt the acoustic signal as the main medium of input interface, while leveraging the wireless signal to recognize the user context and initiate the communication between devices. Second, we show the potential of the new methods by taking an application-driven approach in spontaneous and real-world scenarios. We believe that our proposals can compensate the limitation of existing interaction solution, which can be used simultaneously to improve the user experience. As a basic enabling technology that builds up the foundations of the mobile computing, we expect that this dissertation can enable seamless integration of the
new technology.

1.1 2D and 3D Gesture Tracking for Mobile Devices

Although touchscreens on mobile devices allow intuitive interactions through haptic communication, the user experience is limited by the size of two-dimensional touchscreens. By adopting audio sensors such as a speaker and multi-channel microphones that are already embedded on mobile devices by default, we extend the range of mobile interaction from touchscreens to any surfaces near the device. Additionally, we stretch the scope from a two-dimension surface to a three-dimensional space.

In Chapter 2, we propose vTrack [9], a virtual trackpad interface that tracks the user's input movements near the mobile device. We build our system based on the sound source localization technique using the acoustic signal for interaction, because it does not require any expensive sensors or additional resources on mobile devices. We demonstrate that various characteristics of audio signal can be applied in the context of mobile interaction and propose a novel coordinate positioning algorithm that achieves millimeter-level tracking accuracy.

In Chapter 3, we extend the vTrack system to track the sound source in a three dimensional space. Our audio-based 3D tracking approach compensates the limitation of the vision-based motion tracking, which has a limited field of view confined by the camera's angle. By adopting additional cues including the inverse distance law and Doppler effect of the sound, we reduce the ambiguity in inferring the depth information of the sound source in a three dimensional space and accomplish continuous 3D motion tracking with high-accuracy.

1.2 Unobtrusive Interaction Interface for Multiple Co-located Devices

The proliferation of pervasive devices is offering great opportunities to enhance user experiences through the network of devices. Heartbeats read by a smart watch can be monitored on the user's smart devices, and photos taken by smart glasses can be shared with the user's digital frame at home. We focus on the prospect of high-density ubiquitous devices and prevalent Wi-Fi networks in indoor environments.
Customary device configuration for establishing connection among devices usually manipulate the device identifier uniquely associated with the recipient. These methods involve mundane steps of searching and specifying the receiver, which consume both considerable time and effort to initiate the communication. It definitely degrades the handiness of the service, and will increase nuisances specifically when used by ubiquitous devices that incur frequent networking. This induces the need of a new interaction interface design.

Chapter 4 continues our study by presenting Flock [8], an unsupervised proximity detection and pairing system for smart devices that instantly discovers and connects devices by sensing their physical proximity. Using Flock, users can automatically discover and interact with others in the vicinity without explicitly searching and specifying the receivers’ device. Proximity detection is accomplished through the cloud server that collects a series of Wi-Fi scan results from each user device and matches their received signal strength (RSS) patterns using a similarity metric, which returns quantitative estimates of relative distance measure.
Chapter 2

A Virtual Trackpad Interface using Sound Source Localization

Touchscreens on mobile devices allow intuitive interactions through haptic communication, but their limited workspace confines user experiences. In this chapter, we envision a virtual trackpad interface named vTrack that tracks user input on any surface near the mobile device. We adopt the acoustic signal for the interaction, which can be handled by lightweight signal processing using inexpensive sensors on mobile devices. In our vTrack prototype, the peripheral device simply emits inaudible acoustic signals through a loudspeaker, while the receiving device performs sound source localization by leveraging a multi-channel microphone array. We build a fingerprint-based localization model using various cues, such as time difference of arrival, angle of arrival, and power spectrum density of the audio signal. The vTrack system integrates the frequency difference of arrival incurred by the Doppler shift to track the sound source in motion. Finally, the position estimations are fed into the extended Kalman filter to reduce errors and smooth the output. We implement our system on Android devices and validate its feasibility. Our extensive experiments show that vTrack achieves millimeter-level accuracy in the moving sound source scenario.

2.1 Introduction

Although the size of touchscreens on mobile devices is getting bigger [1], the input interface of mobile devices still limits user experiences in mobile interactions. Software keyboards toggled for text input are error-prone due to the fairly small key size compared to human
fingers. The major inconvenience incurred by the touch input is the finger itself, which blocks the screen during the interaction. Therefore, we focus on the aspect that any surface near the mobile device can operate as a virtual trackpad, if user’s movement can be accurately traced. Using this virtual trackpad, users can manipulate the mobile device from outside the touchscreen, without blocking the view of the screen; for example, they can rewind or fast-forward a video while playing it or control characters during the mobile games.

In this chapter, we envision vTrack, a virtual trackpad interface that tracks the user’s input near the mobile device, which extends the range of mobile interaction over the touchscreen. We build the system based on sound source localization technique using the acoustic signal, because it neither requires expensive sensors nor computation-intensive processing on mobile devices. In addition, our system design does not utilize explicit networking between the peripheral and receiving device, because the peripheral does not transmit any data to the receiver; rather, it only emits the sound signal. This excludes any networking module on the peripheral and allows the system to maintain as minimal resource as possible. Low computing and communications on device consequently results in low power consumption, which meets the most significant requisite of peripheral devices.

For the proof-of-concept, we use an off-the-shelf mobile device as the peripheral, which functions as an input device held by a user. Furthermore, we design and build a pen-shaped peripheral using a 3D printer to enhance the usability. In further stages of production, any type of wearable devices with a speaker including smart watches and rings may adopt vTrack to allow ubiquitous interactions with mobile devices. As the peripheral is equipped with a speaker that only costs a few dollars, we expect drastic reduction of the product price when commodified.

The receiving device detects the audio signal and performs sound source localization on-device in real-time. The signal processing and localization procedures are entirely managed by the receiver by taking advantage of its relatively abundant computing resources compared to the peripheral. It uses three-channel built-in microphones that come with a modern smartphone: a primary microphone is for voice recording purposes, while other two are intended to perform stereo recording and cancel the background noise. Figure 2.1 shows vTrack’s operation scenario using two mobile devices. An external speaker peripheral is connected to the transmitter device.

There exist several challenges in envisioning the vTrack system. The key challenge is to achieve fine-grained localization accuracy with restricted resources on the peripheral side.
Our assumption that the peripheral device has an asymmetric capability compared to the receiver device differentiates our work from others in the literature. It limits the system to perform one-way sensing in a synchronization-free manner, with no additional cues for localization other than the sound signal. Second, the audio sampling rate is bounded to 192kHz at driver- or operating system-level on mobile devices, which limits the positioning granularity to 1.8mm when solely relying on the time difference of audio samples. Finally, the moving sound source increases the uncertainty involved in the acoustic sensing, due to various reasons such as multipath reflections and variance in the signal direction.

In this chapter, we make contributions by demonstrating that various principles related to the sound source localization technique can be applied in the context of mobile devices and are feasible for positioning the sound source near the mobile device with millimeter-level accuracy. We build a fingerprint-based sound source localization model using the Time Difference of Arrival (TDoA), Angle of Arrival (AoA), and Power Spectrum Density (PSD) of the audio signal. To track the moving sound source with high precision, we leverage Frequency Difference of Arrival (FDoA) incurred by the Doppler shift, and feed the movement direction and velocity measurements to the extended Kalman filter, which reduces the effect of uncertainties in the position estimation. We apply noise reduction technique during the signal processing to prevent the interference from high-frequency noise and improve the
We have implemented vTrack using two Android devices. Our experiment verifies that combinations of the aforementioned cues are sufficient to resolve the challenges and envision sound source localization with high accuracy. The AoA of the sound measurements are also noticeable: 0.8° of angle estimation error and 1.1mm of range estimation error. In general, vTrack achieves 1.1mm positioning accuracy in moving sound source scenario, which surpasses the theoretical granularity limited by current audio sampling rate. These promising results demonstrate the potential of vTrack in sound source localization-based mobile interactions.

This chapter is organized as follows. Section 2.2 reviews previous work in sound source localization and mobile interaction literatures that are closely related to the proposed system. Section 2.3 describes the underlying core ideas of our system and validates its feasibility. Section 2.4 discusses signal processing and implementation details, and Section 2.5 presents performance of the system based on our extensive experiments. Finally, Section 2.6 concludes the chapter.

2.2 Related Work

In this section, we first summarize related work that performs sound source localization in different scenarios. Then, we introduce various near-device mobile input techniques proposed in the literature, as well as commercial devices in the market.

2.2.1 Sound Source Localization

There exists significant research on acoustic signal-based positioning techniques. We categorize these techniques into two groups by the scope of their target scenario.

2.2.1.1 Large-Scale Localization Scenarios

BAT [18] is an indoor localization system, which is based on the time of flight measurement of ultrasound signals. Similarly, Cricket [44] adopts ultrasound in combination with the RF signal. It infers the distance through time difference of arrival of the concurrent transmissions of both signals. These systems achieve a centimeter-level resolution, but require densely deployed infrastructural supports consisting of specially designed ultrasound transceivers.
Recently, peer-assisted acoustic ranging scheme [29] is proposed to improve the localization errors induced from the intrinsic limit of the RF signal propagation. Also, PANDAA [52] determines the relative locations of networked sensors measuring the time difference of arrivals of ambient sound in the room. Aforementioned systems aim to localize the object in large-scale indoor scenarios (i.e., building or room-size environments), while our operation space is bounded to tablet-size range.

Other acoustic localization schemes mainly targeted for outdoor environments include ENSBox [12], which is an angle of arrival-based distributed system integrated on ARM platform with four-channel microphones at each node. By virtue of its relatively abundant microphone array that is geometrically arranged in 3D space, it achieves high accuracy in object positioning, up to few centimeters. Whistle [59] leverages the time difference of arrival of the acoustic sound observed at different receiving devices, which serve as the basic infrastructure of the system. Finally, [61] classifies the position of device in a car to detect driver phone use by analyzing the TDoA of acoustic signals emitted from four-channel built-in speakers.

2.2.1.2 Small-Scale Localization Scenarios

Our work is closely related to proposals that study ranging between nearby devices. BeepBeep [40] is an acoustic ranging mechanism that operates on off-the-shelf mobile devices. Two devices emit a beep sound in turn and simultaneously record both beep sounds. From the (self) audio recording, each device can measure the elapsed time between two beep sound events. By exchanging this time information, devices can obtain the time of flight of the two beeps, and consequently get the distance between two devices. BeepBeep assumes two mobile devices to have equivalent computing capability, and coordinated with wireless communications such as Wi-Fi or Bluetooth. Its ranging accuracy exceeds one centimeter, which is inappropriate for applications that require precision.

Based on the BeepBeep procedure, [45] extends the phone-to-phone localization to three-dimensional space. By leveraging multiple microphones and other inertial sensors on mobile devices, it performs 3D triangulation using time of arrival and signal power of the acoustic signal. Through continuous localization, each device can estimate the other device's relative position and track its movement. However, this work aims for a meter-level operation range, and achieves 3D localization accuracy with several centimeters of position error. Similarly, [64] proposes a high-speed acoustic ranging scheme in 3D space for fast moving mobile
devices, which enables phone-to-phone motion games.

2.2.2 Mobile Input Techniques

Various types of mobile input methods have been proposed recently, which refrain from direct interactions with the touchscreen due to the inherent limitations of touch interface. UbiK \[55\] is a portable text-entry method that requires a keyboard outline printed on a paper. It makes use of the dual-microphone interface on a mobile device to localize the keystroke sound on solid surfaces such as desk. UbiK copes with the acoustic multipath fading through Amplitude Spectrum Density (ASD) of different keystroke sound, and localizes distinct keystroke locations by fingerprinting-based signature matching. This scheme is highly dependent upon the surface environments it works on, and demands of repetitive training process every time when the workspace changes. Similarly, [30] snoops keystrokes behind the keyboard based on the TDoA and other acoustic features.

Okuli \[63\] takes Visible Light Sensing (VLC) technology into account to let the VLC-capable mobile device to sense the movement of user’s finger. It extends the mobile interaction workspace to nearby surfaces and enables a virtual trackpad and keyboard input. Using an LED transmitter and two photodetectors as peripherals, Okuli builds a model-driven framework based on the physical properties of the visible-light channel for finger positioning. It achieves around one-centimeter scale precision, but requires a specially designed peripheral equipped with additional sensors. Other than visible light, the uses of RFID tag and receivers \[54\] and inertial motion sensors \[2, 60\] for handwritten text in the air have been proposed for mobile input.

Smartpen (i.e., digital pen) devices available on the market adopt various technologies to capture the user’s path of movement. Livescribe \[31\] smartpen is equipped with an infrared camera at the tip that determines its position on the page when used with the paper pre-printed with a dot pattern. To record the input, the user should write on this special paper, which leverages the dot-positioning system. Equil \[10\] pairs with its receiver, which uses both ultrasound and infrared to accurately locate the smartpen’s position on the paper. By attaching the receiver to the top of the paper, the user can take notes on any paper surface. Phree \[42\] turns any surface into a virtual canvas by adopting a 3D laser interferometer and an optical sensor, in addition to motion-capturing sensors embedded in the device. Smartpens usually process the sensor readings on-device and send the interpreted data to the paired mobile device over networks such as Bluetooth. The limitation of smartpens is
their high price range—from $100 to $200—due to the adoption of various expensive sensors and computing resources to achieve a high level of accuracy when determining the position of the pen.

2.3 2D Sound Source Tracking

vTrack is composed of two mobile devices as illustrated in Figure 2.1. Receiving device is placed horizontally to perform audio signal processing and display the cursor on screen along with the user’s movement. The peripheral periodically emits audio signals, and is allowed to move around on the virtual trackpad printed on a piece of paper, which is a 10×13 grid with 1cm unit. We verify later through experiment that the size of the virtual trackpad is not necessarily limited to the distance between microphones, but can be enlarged to twice its current size.

2.3.1 Coordinate Positioning

2.3.1.1 Time Difference of Arrival

Multilateration is a common navigation technique based on the measurement of the difference in distance to two known locations. Because it utilizes the relative difference instead of the absolute distance measures, it is free from clock synchronization between the signal transmitter and its receiver. We benefit from this property to build a synchronization-free one-way sensing system, and use the Time Difference of Arrival (TDoA) of sound for coordinate localization. For TDoA computation, we adopt Generalized Cross Correlation with Phase Transform (GCC-PHAT) algorithm [24], which is known to effectively reduce the noise and reverberation and perform well in actual noisy environments [5]. The index count of the maximum absolute value of GCC-PHAT is considered as the time lag between two audio signals.

The first graph in Figure 2.2 presents the normalized TDoA value for two audio signals recorded at microphones 1 and 2 in Figure 2.1. TDoA values gradually increase as the sound source moves from left to right. Figure 2.3a shows the cumulative distribution function (CDF) of cross-correlation index values that are repetitively collected for 50 times on each column of a certain row. Most of the measurement points have values oscillating within the range of two to three sample indexes, which shows high stability. Dotted lines represent another
Figure 2.2 Normalized time difference of arrival (top) and power ratio values (middle and bottom) of audio signals recorded by two microphones. Measurements are taken at each point of the grid.

measurement results collected on a different day, and we can observe that cross-correlation index values at the same point are consistent over time and thus reproducible.

Based on this consistency, we first perform linear regression on the measurement data and model the relationship between TDoA and the x-coordinates (i.e., left-right). However, because TDoA is a relative difference measure, locations that have a constant value form a hyperbolic curve, which result in modeling ambiguity. To locate the exact position on the hyperbola, we introduce the third microphone embedded at the bottom of the mobile device. The second measurement taken by a different pair of microphones (i.e., microphones 1 and 3) will produce the second curve, which intersects with the first one. When the two curves are compared, a small number of possible locations are considered as candidate positions of the sound source.

Although three-channel microphones reduce the search space, the positioning resolution is limited by the audio sampling rate. TDoA resolution is 0.005ms with 192kHz sampling rate. When we convert it to distance, the resolution becomes 1.8mm. Considering the uncertainties
Figure 2.3 Distribution of cross-correlation index value and power ratio of audio signals measured on two different days (solid lines: first day, dotted lines: second day).

included in the measurements, deterioration of positioning accuracy is inevitable. To handle this, we introduce additional cues from audio signal characteristics.

2.3.1.2 Power Spectrum Density

Total power level of the signal tends to increase as the sound source approaches the receiver and decrease during the recession. However, as the microphone is usually located in the middle of edges and embedded perpendicularly to the edge, the change in power level does not show linear relationship with the y-axis (i.e., front-back). When the sound source recesses from the receiving device along the y-axis, there exists a non-monotonic section. In addition, we observe that sensitivity of the power level at two microphones highly differs. Because minor microphones are designed for background noise canceling, it performs better than the major microphone in detecting the sounds with high frequency at a distance. Contrarily, the major microphone is targeted for voice signals in short range, so it cannot sensitively detect high frequency sound signals that are far away from it.

From our extensive experiments, we observe that the amplitude of signal emitted by the transmitter is not consistent throughout the experiment session due to the hardware limitation, although the volume of the sound is set to constant value. Even small variation in signal power can be interpreted as error in modeling the distance to the sound source using
the absolute signal amplitude. Therefore, we decide to adopt the ratio of the power level at microphones, because the ratios remain consistent even the absolute power may vary. Lower two plots in Figure 2.2 present the power ratios of three microphones.

To verify the temporal stability of power ratio at each point of the grid, we measure the power readings on two different days. Figure 2.3b shows the CDF of power ratio of two microphones that are repetitively collected for 50 times on each row of the same column. Solid lines represent results from the first session and dotted lines stand for the second session. Distribution of the power ratio at each point remains the same for different measurement sessions, which shows the temporal stability of power ratio.

After collecting the cross-correlation value and power ratios of three microphones at each point of the grid, we construct a database that consists of five-tuple data:

$$[x_{corr12}, x_{corr13}, x_{corr23}, \frac{mic1}{mic2}, \frac{mic3}{mic1}]$$

where \(x_{corr12}\) stands for the cross-correlation value between microphones 1 and 2. To find the best estimates of x- and y-coordinate of the input data, we perform k-nearest neighbor (KNN) search on the database using the Euclidean distance. To make the matching algorithm robust to abrupt erroneous readings, we use the moving average of input data. We also keep track of the previous states as a reference, and intelligently filter out unexpected errors.

However, some points have overlapping range of power ratio, which results in ambiguity in differentiating two points. For example, two neighboring points (2,2) and (2,3) as well as points (2,7) and (2,8) in Figure 2.3b have power ratio distribution that are very close to each other. This is due to the refraction of sound signal by the corner of the device in addition to the multipath reflections. Thus, we adopt the angle of arrival of the signal and integrate it in the positioning algorithm to correct possible errors.

2.3.1.3 Angle of Arrival

First, we start with a scenario using two microphone array. Consider a \(\triangle ABC\) between the receiving device and the sound source as shown in Figure 2.4, where AoA is \(\theta\) by assuming that the microphone is located at the corner of the device. By applying the cosine rule for
Figure 2.4 Angle of arrival ($\theta$), range ($r$), and perpendicular distance ($y$) estimation using the time difference of arrival ($\Delta t$) measure at two microphones.

\[ \triangle ABC \], we get

\[
\cos \theta = \frac{l^2 + r^2 - (r + \Delta t \cdot c)^2}{2l \cdot r} \tag{2.1}
\]

where:

- $l$ = Distance between two microphones
- $r$ = Shorter distance from sound source to microphone
- $\Delta t$ = Time difference of arrival at two microphones
- $c$ = Speed of sound in dry air at 20°C (343.2 m/s)
- $x$ = Offset of the sound source from the left edge.

We can compute the range $r$ and the angle of arrival $\theta$ using TDoA, and consequently get the distance $y$ of the sound source to device as follows.

\[
\begin{align*}
    r &= \frac{l^2 - (\Delta t \cdot c)^2 - 2l \cdot x}{2\Delta t \cdot c} \tag{2.2} \\
    \theta &= \arccos \left( \frac{x}{r} \right) \tag{2.3} \\
    y &= r \cdot \sin \theta \tag{2.4}
\end{align*}
\]
Figure 2.5 Coordinate estimation of the sound source \( S(X_s, Y_s) \) using the distance to three microphones \( M1, M2, \) and \( M3 \) presented on a Cartesian coordinate system.

### 2.3.1.4 Multilateration

Based on our \( x \) and \( y \) coordinate estimation using two microphones, we extend the method by including additional information from the third microphone, which contributes to improve the positioning accuracy. Because handling a set of nonlinear hyperbolic equations is challenging on resource-constrained mobile devices, we derive the sound source position as follows. By assuming the receiving device is located in a coordinate system as shown in Figure 2.5, coordinates of three microphones can be represented as \( M1, M2, \) and \( M3, \) respectively. Then, the distance from the sound source to each microphone is formulated as follows:

\[
\begin{align*}
    d_1 &= \sqrt{(X_s - X_1)^2 + (Y_s - Y_1)^2} \\
    d_2 &= \sqrt{(X_s - X_2)^2 + (Y_s - Y_2)^2} \\
    d_3 &= \sqrt{(X_s - X_3)^2 + (Y_s - Y_3)^2}
\end{align*}
\]  

(2.5)

By subtracting the square of distances \( d_1 \) and \( d_2, \) we get

\[
d_1^2 - d_2^2 = X_s^2 - X_2^2 - 2X_s(X_1 - X_2) - 2Y_s(Y_1 - Y_2) + Y_1^2 - Y_2^2 
\]  

(2.6)

which can be rearranged as follows:

\[
2X_s(X_1 - X_2) + 2Y_s(Y_1 - Y_2) = X_1^2 - X_2^2 + Y_1^2 - Y_2^2 - d_1^2 + d_2^2
\]  

(2.7)
Similarly, relationship between \(d_1\) and \(d_3\) becomes

\[
2X_s (X_1 - X_3) + 2Y_s (Y_1 - Y_3) = X_1^2 - X_3^2 + Y_1^2 - Y_3^2 - d_1^2 + d_3^2
\]  \quad (2.8)

Then, these relationships can be formulated in matrix form

\[
2A \begin{bmatrix} X_s \\ Y_s \end{bmatrix} = B
\]  \quad (2.9)

where:

\[
A = \begin{bmatrix} X_1 - X_2 & Y_1 - Y_2 \\ X_1 - X_3 & Y_1 - Y_3 \end{bmatrix}
\]

\[
B = \begin{bmatrix} X_1^2 - X_2^2 + Y_1^2 - Y_2^2 - d_1^2 + d_2^2 \\ X_1^2 - X_3^2 + Y_1^2 - Y_3^2 - d_1^2 + d_3^2 \end{bmatrix}
\]  \quad (2.10)

The distance difference between microphones \(\Delta d_{21}\) and \(\Delta d_{31}\) can be represented using the TDoA measures.

\[
\Delta d_{21} = d_2 - d_1 = t_2 \cdot c - t_1 \cdot c = \Delta t_{21} \cdot c
\]

\[
\Delta d_{31} = d_3 - d_1 = t_3 \cdot c - t_1 \cdot c = \Delta t_{31} \cdot c
\]  \quad (2.11)

As the range \(d_1\) is known from the Equation 2.2, the distances \(d_2\) and \(d_3\) are obtained. Therefore, the matrix B becomes

\[
B = \begin{bmatrix} X_1^2 - X_2^2 + Y_1^2 - Y_2^2 + \Delta d_{21}^2 + 2\Delta d_{21} d_1 \\ X_1^2 - X_3^2 + Y_1^2 - Y_3^2 + \Delta d_{31}^2 + 2\Delta d_{31} d_1 \end{bmatrix}
\]  \quad (2.12)

Finally, by projecting the microphone coordinates on the Cartesian coordinate system, we substitute the coordinates \((X_1, Y_1) = (0, 0), (X_2, Y_2) = (l, 0),\) and \((X_3, Y_3) = (l, h),\) and get matrices A and B,

\[
A = \begin{bmatrix} l & 0 \\ l & h \end{bmatrix}
\]

\[
B = \begin{bmatrix} l^2 - \Delta d_{21}^2 - 2\Delta d_{21} d_1 \\ l^2 + h^2 - \Delta d_{31}^2 - 2\Delta d_{31} d_1 \end{bmatrix}
\]  \quad (2.13)
where:

\[ l = \text{Distance between microphones 1 and 2} \]

\[ h = \text{Distance between microphones 2 and 3}. \]

Now, we can infer the position of the sound source using the TDoA measurements with three microphone array. However, in the moving sound source scenario, the result may be erroneous due to the unexpected deviation in the direction and/or angle of the speaker. Therefore, we adopt additional cues that can be obtained from the moving sound source.

2.3.2 Movement Detection

2.3.2.1 Frequency Difference of Arrival

Doppler effect refers to the phenomenon where the frequency of a sound wave increases as the sound source approaches observer, and decreases during the recession. To detect the
movement of the sound source relative to the observer, we leverage the change in frequency of the sound wave. When the sound source approaches the observer with velocity $v_s$ during the period $T$ of the sound wave, the wavelength becomes $\lambda' = \lambda - v_s T$ as the sound source gets closer as $v_s T$. By substituting the equation with properties $\lambda' = v / f'$, $\lambda = v / f$, $T = 1 / f$, and $v$ with speed of sound $c$, we get the observed frequency $f'$.

\[
f' = \left( \frac{c}{c - v_s} \right) f
\]

(2.14)

where $f$ is the original frequency emitted by the transmitter. Thus, we can compute the velocity of the sound source $v_s$:

\[
v_s = c \left( 1 - \frac{f}{f'} \right)
\]

(2.15)

where $v_s$ is positive when the sound source approaches the receiver and negative in the other direction. As we know the velocity of the sound source, we can also derive the amount of movement by multiplying $\Delta t$, which can be known from the timestamp.

As an example, we measure the change in frequency and received power level at each microphone when the sound source moves along the path connecting points $(x, y) = (4, 3) \rightarrow (4, 8) \rightarrow (9, 8) \rightarrow (9, 3) \rightarrow (4, 3)$ in the grid sequentially, drawing a rectangle clockwise. Because the first movement $(4, 3) \rightarrow (4, 8)$ puts the sound source away from both microphones, frequency readings show abrupt decrease at both microphones in Figure 2.6. Similarly, the third movement where the sound source approaches both microphones makes the frequency increase at both microphones. We can observe that when the sound source moves along the y-axis, the rise and fall of frequency at both microphones coincide. Contrarily, frequency changes at two microphones are opposite when the sound source follows the x-axis direction. By leveraging this frequency difference of arrival, we can infer the direction of the sound source movement.

Changes in the sound amplitude observed at each microphone can also assist the movement detection. The amplitude tends to increase as the sound source moves toward the microphone, and decrease when moving in the reverse direction. The second figure in Figure 2.6 presents the sound pressure, which is derived from the measured amplitude value in decibel ($\text{dB} = 20 \log \frac{p}{p_0}$) by assuming the reference sound pressure $p_0$ as 0.00002Pa. We can observe the asymmetric performance of two microphones. Microphone at the bottom of the device (mic 2) shows linear amplitude transition as a function of distance. However, microphone at the top (mic 1) responses with high sensitivity to the sound in short distance, while the re-
sponse sensitivity sharply decreases after some distance. Although there exists ambiguity with long distance, each movement to different direction is differentiable with the increase and decrease of amplitude as follows: \( \downarrow = [\text{decrease, decrease}] \), \( \rightarrow = [\text{decrease, increase}] \), \( \uparrow = [\text{increase, increase}] \), and \( \leftarrow = [\text{increase, decrease}] \), where the arrows represent the movement direction and the tuples stand for the amplitude change of \([\text{mic}_1, \text{mic}_2]\).

Finally, we can now use the aforementioned cues regarding the direction of the sound source to assist our decision making process. We can compute the velocity of the moving sound source by using the frequency difference as shown in the bottom figure of Figure 2.6, and consequently calculate the distance of movement by taking the integral of the plot. In this specific example, each movement involves 7 samples each, where the sample interval is set to 300ms. Thus, the amount of displacement can be approximated to 4.2cm, which is close to the ground truth.

### 2.3.2.2 Extended Kalman Filter

Due to the noisy measurements induced by the movement, the uncertainties in inferring \(x\) and \(y\)-coordinates are unavoidable. To filter out erroneous readings and smooth the output, we introduce extended Kalman filter, which is a Markov model that assumes dependence on the previous state only. The filter deals with uncertain measurement about the dynamic system that is continuously changing, and makes an educated estimation about what the next state will be. It is widely used in technology such as navigation and control of vehicles, time series analysis in signal processing, and robotic motion control and trajectory optimization. In the prediction step, the filter produces estimates of the current state taking uncertainties into account. Once new measurement is observed, estimates are updated using a weighted average, with more weight being given to estimates with higher certainty.

The state vector \(x_k = [p_x, v_x, p_y, v_y]^\top\) in our model involves estimating not only the \(x\) and \(y\) coordinates but also its \(x\) and \(y\) velocities. These four states must be estimated given only noisy measurements of range and angle. The measurement vector \(z_k = [r, \theta]^\top\) contains the actual range and angle readings as illustrated in Figure 2.4. The state transition and measurement models are represented as follows:

\[
\begin{align*}
x_{k} &= f(x_{k-1}) + w_k \\
z_{k} &= h(x_{k}) + v_k
\end{align*}
\]

(2.16)

where \(w_k \sim \mathcal{N}(0, Q_k)\) and \(v_k \sim \mathcal{N}(0, R_k)\) are the process and measurement noises, which
are both assumed to be zero mean multivariate Gaussian noises with covariance $Q_k$ and $R_k$, respectively.

The process noise matrix $Q_k$ measures the variability of the input signal away from the ideal transitions defined in the state transition matrix. Larger values in this matrix mean that the input signal has greater variance and the filter needs to be more adaptable. Smaller values result in a smoother output, but the filter is not as adaptable to large changes. In our model, we define $Q_k$ through some fine-tuning process. The measurement noise matrix $R_k$ defines the error of the measuring device. We determine this accuracy empirically as well. Decreasing the values in this matrix means we are optimistically assuming our measurements are more accurate, so the filter performs less smoothing and the predicted signal will follow the observed signal more closely. Conversely, increasing the values means less confidence in the accuracy of the measurements, so more smoothing is performed.

The state transition function $f$ computes the predicted state from the previous estimate. Similarly, the measurement function $h$ computes the predicted measurement from the predicted state. As the displacements and velocities are non-linearly related to the range and angle, the filter algorithm requires calculation of a matrix of partial derivatives (the Jacobian) for the state and measurement equations. So, in predict stage, predicted state $\hat{x}_k$ and predicted covariance $P_k$ are represented as follows. $F_k$ is the Jacobian matrix of state transition function.

$$\hat{x}_k = F_k \hat{x}_{k-1}$$
$$P_k = F_k P_{k-1} F_k^\top + Q_k$$

(2.17)

The measurement update equation uses the range and angle, which are related to the $x$ and $y$ displacements as shown in Equations (2.2)-(2.4). The Jacobian matrix $H_k$ for the measurement equations is computed as follows.

$$h(x_k) = \begin{bmatrix} r_k \\ \theta_k \end{bmatrix}$$

(2.18)

$$H_k = \left. \frac{\partial h(x_k)}{\partial x_k} \right|_{\hat{x}_k}$$

(2.19)

The update stage takes the measurement $z_k$ and computes the measurement residual $\tilde{y}$ and
covariance $S_k$, and gets the Kalman gain matrix $G_k$.

$$\bar{y} = z_k - h(\hat{x}_k)$$
$$S_k = H_k P_k H_k^\top + R_k$$
$$G_k = P_k H_k^\top S_k^{-1}$$

Finally, we get our new best estimates for state $\hat{x}_k'$ and covariance $P_k'$.

$$\hat{x}_k' = \hat{x}_k + G_k \bar{y}$$
$$P_k' = P_k - G_k H_k P_k$$

The filter recursively iterates these two stages and updates the position estimation of the moving sound source.

### 2.4 Implementation

We implement the proposed model using Android devices and run the sound source localization algorithm in real-time. Samsung Galaxy Note 4 is used as a receiver, while Samsung Galaxy S II operated as the audio signal transmitter. As the Android OS does not provide APIs that can handle three-channel audio streams by default, we modify the Android audio framework system to deliver raw PCM data generated at the audio hardware to the application layer. The process of vTrack system is illustrated in Figure 2.7 as a flowchart.

#### 2.4.1 Audio Signal Design

As we neither use time synchronization method to synchronize two devices nor exchange any informative data between them, the audio signal is the only mean that can control our system. Thus, the signal and its protocol should be selected with care. According to our experiment, Samsung Galaxy Note 4 has high quality frequency response ranging from 20Hz up to 20kHz. We choose the frequency range of the audio signal to be 19kHz, which is almost inaudible. It does not generate any disturbing sound during the operation, and also can be used in usual environment without interference because it does not overlap with the frequency range of voice speech.

The receiving device continuously reads audio data from its audio hardware in chunks, so
Figure 2.7 vTrack’s operation flowchart on the receiving mobile device. Audio signal processing module performs frequency-domain analysis on the audio signal. Coordinate computation module obtains various cues from the signal for coordinate positioning.

the signal emitted by the transmitter should fit into the buffer size with good cross-correlation property. An audio signal that is too short may not be clearly distinguishable with the ambient noise. On the other hand, a longer signal consumes excessive time in reading and processing the audio data, which incurs delay in the update interval. Based on our extensive experiments, we decide to generate a 2ms-long sine tone signal shown in Figure 2.8 periodically with 21ms interval, which shows recognition rate over 98%.

2.4.2 Signal Processing

We choose the size of buffer that system periodically reads audio data in and the period of audio sound to be the same, so that each audio chunk contains one audio sample. Aligning the buffer size and the audio signal period does not support perfect synchronization between the transmitter and receiver. Although the system issues a command to play an audio out
at a certain time, there exists some delay in time when the signal is actually emitted. This latency in audio playback is still an open issue in Android OS. Thus, the clock drift between two devices is inevitable. To guarantee the signal detection in every chunk of audio data, we force shift the starting index of buffer. When there is more than one peak that is larger than the threshold value, we read the next signal data from the buffer starting from index increased by half of the buffer size.

To detect FDoA at two microphones, the received audio signal should be transformed from time domain into frequency domain. Representing the given signal in frequency domain is usually done via Fast Fourier Transform (FFT), which implements Discrete Fourier Transform (DFT) in an efficient manner. Power spectrum is desired for analysis in frequency domain, where the power of each frequency component of the given signal is plotted against their respective frequency. We compute $P_x(f) = X(f)X^*(f)$, where $X(f)$ is the frequency domain representation of the signal $x(t)$, and $X^*(f)$ is the complex conjugate of $X(f)$.

However, Fourier transform is known to return spectrum of an entire sequence, which is not appropriate to analyze the time-varying signals. As the received spectrum is non-stationary due to the fast-moving sound source, we adopt Short Time Fourier Transform (STFT) technique to capture the frequency shift. STFT segments the received signal into narrow time intervals (i.e., window) and takes the Fourier transform of each segment in chunk, providing simultaneous time and frequency information.

The window size should be narrow enough to ensure that the portion of the signal falling within the window is stationary. But very narrow windows do not offer good localization in

![Figure 2.8 Sample waveform of 2ms-long sine signal emitted every 21ms by the transmitting device.](image)
the frequency domain. When the signal is sampled at a sampling rate $f_s$ over an acquisition time $T$, $N$ samples are acquired according to the following equation.

$$T = \frac{N}{f_s} \quad (2.22)$$

The frequency resolution $\Delta f$ is then determined by the acquisition time.

$$\Delta f = \frac{1}{T} = \frac{f_s}{N} \quad (2.23)$$

Frequency resolution is the ability to differentiate two closely spaced signals. It improves as the acquisition time increases. At a fixed sampling rate, increasing the frequency resolution decreases the temporal resolution. This inverse relationship between time and frequency resolution directly affects the performance of our system: it needs to achieve a precise frequency resolution to detect movement related cues through the frequency difference and a high time resolution for system responsiveness at the same time. Therefore, we empirically choose $N = 12,288$ samples as the FFT length, which allows to retrieve data at 33Hz rate as well as a sufficient frequency resolution for real-time motion tracking.

We implement STFT procedure that returns a Power Spectrum Density (PSD) array of the received signal. From the output PSD array, we extract the maximum value where its frequency is within the frequency threshold. In order to get the TDoA between two signals $x_i(t)$ and $x_j(t)$ at each microphone, the GCC-PHAT is computed as follows:

$$G_{PHAT}(f) = \frac{X_i(f)X^*_j(f)}{|X_i(f)X^*_j(f)|} \quad (2.24)$$

and the time difference for these two microphones is estimated:

$$d_{PHAT}(i, j) = \arg\max_{d} (R_{PHAT}(d)) \quad (2.25)$$

where $R_{PHAT}(d)$ represents the inverse Fourier transform of the function $G_{PHAT}(f)$. The audio sampling rate at the receiver is set to $f_s = 192$kHz, which is the highest value supported by modern Android devices. By multiplying the cross correlation value with the period of sample, we can compute TDoA of sound signals captured at two microphones as follows.

$$\Delta t = \frac{1}{f_s} \times \Delta I \quad (2.26)$$
2.4.3 Noise Reduction

To adaptively handle the background noise and improve the positioning accuracy, we adopt noise reduction technique. First, we apply band-pass filter to the frequency-domain audio signal to remove noise at unintended frequencies. As we use the audio signal at 19kHz frequency, we pass the frequency band between 18.5kHz and 19.5kHz, leaving upper and lower 500Hz band for FDoA scheme. Applying the band-pass filter effectively blocks most of the background noise and leaves the frequency band sent by the transmitter. However, when there are high frequency noises such as machine sound and fan noise, the band-pass filter performs poorly and degrades the positioning performance. Figure 2.9 illustrates the frequency-domain audio captured at a cafe. It contains high frequency noise around 19kHz, which significantly interferes the detection of our audio signal from the peripheral. Therefore, we implement a noise reduction technique to extract clean original signal.

When the system starts, we record the ambient sound for a short period of time and build a noise model for current background noise. At this time, the peripheral remains silent. Using the noise model, the receiver performs spectral subtraction to the incoming audio when the peripheral starts to emit the signal. Finally, the original audio signal is reproduced after the phase recovery and transforming the frequency-domain signal to time-domain signal. The noise reduction procedure is shown in Figure 2.10.

Figure 2.9 Ambient noise measured at a local Starbucks cafe. Various machine sounds result in a high-frequency (~19kHz) background noise.
Figure 2.10 vTrack’s noise reduction procedure builds a noise model by recording the background noise and performs spectral subtraction on the original audio signal through frequency-domain analysis.

2.5 Performance Evaluation

2.5.1 Experiment Setup

We use two off the shelf mobile devices without any modification on hardware for evaluation. We adopt a device with large screen with high screen-to-body ratio as a receiver: a Samsung Galaxy Note 4 which has 5.7 inch display with 1440×2560 pixels on a 153.5×78.6×8.5mm body. Microphones 1 and 2 are aligned horizontally, while microphones 2 and 3 are aligned vertically along the bottom edge with 33mm spacing as shown in Figure 2.1. The receiving mobile device continuously displays a cursor on the screen, which shows the trajectory of sound source movement.

The audio-signal transmitter, Samsung Galaxy S II, has a speaker at the backside of device. To enhance the usability of the transmitter, we design a pen-type peripheral device that embeds a small speaker at the tip of the pen, and prototype it using a 3D printer. It leverages a 16×12mm speaker with 1.2W rated output power, which is adopted by modern devices including Samsung Galaxy S7. We redirect the output wire of an embedded speaker of the mobile device, which repeatedly generates audio signals and physically connect the output with the peripheral.

We first run experiments in a quiet office room and study the impact of background noise later. We perform operations on a spacious desk with any objects on it in order to minimize the multi-path effects. Our evaluation covers tracking over 5000 of movements composed of straight lines and curves.
2.5.2 Static Sound Source Localization

We analyze the localization performance when the sound source is static. Position error is defined as the Euclidean distance between the estimation result and the ground truth position of the grid.

We first measure the angle and range error of the estimated value. \( \theta \) denotes the angle between the receiving device and the sound source, as illustrated in Figure 2.4. This angle increases from 20° to 70° in 10° intervals. Range \( r \) signifies the displacement of the sound source from microphone 1, assuming that the microphone is located at the corner of the device. The range varies from 6cm to 10cm, in 2cm intervals.

Figure 2.11a plots the measured angle on the designated angle and the range position in the measurement space, where the error bars stand for the standard deviation of the angle error. The average angle error is 0.8°, with a standard deviation of 1.0°. The angle error as well as the standard deviation tends to increase when the angle gets smaller. This is because the effect of refraction on sound propagation increases when the sound source is located at grid points with a small angle. With regard to the measurement range, the average angle error is high (1.2°) in the short range and decreases as low as 0.5° in a longer measurement range. Similarly, Figure 2.11b shows the measured range. The range error also tends to increase with small angles, but the estimation is quite accurate, which shows only a 1.1mm error on
Figure 2.12 Measurement points and results presented using the angle and range values on a polar coordinate system. Gray zone represents vTrack’s trackpad area.

average with a standard deviation of 1.0mm.

Figure 2.12 illustrates the angle and range estimations using the polar coordinate system, where the reference point (0,0) is assumed to be the location of microphone 1 in our scenario. The Cartesian coordinates are also transcribed in parallel for reference purposes. The gray area represents the space where the virtual trackpad targets its operations. We design the dimension of the trackpad to be identical to the actual screen of the receiving device. Due to the bezel on the device’s short edges, there is a 1cm space on the x-axis. The measurement result in the black circle is illustrated with its ground truth measurement point in the white circle, and measurements on the same angle are connected with a white solid line. The error bars stand for the standard deviation of the position error. The measurement result becomes more accurate as the range increases. In the short range, the space between each ground truth is narrower than in the longer range, which accentuates the position error.

Figure 2.13a shows the distribution of the position error on the grid: the average position error is 0.7mm before applying the extended Kalman filter. Specifically, 83% of raw measurements have less than a 1.0mm error. We also measure the position estimation using the extended Kalman filter algorithm and compare the result with the raw data. Because the filter reflects possible errors that may exist in measurements and applies more weight to values with more certainty, it contributes to reducing the position error to 0.3mm on average, with small standard deviations of 0.2mm as well, as shown in the CDF. Over 96% of estimations have errors smaller than 1.0mm, which surpasses the theoretical granularity 1.8mm. By
applying the filter, we achieve a 57% of improvement in the localization performance.

2.5.3 Moving Sound Source Localization

We next measure the performance of localization under the moving sound source scenario. We collect the estimation of the x- and y-coordinates while drawing various paths on the grid space. Four example results are illustrated in Figures 2.14. To derive the position estimation error when tracking the moving source, we compute the shortest distance from each estimation point to the corresponding ground truth path.

Figure 2.13b presents the error distribution of the moving source. The average position error for the raw data case is 1.2mm, with a standard deviation of 0.8mm. It is a slight decrease in the performance compared to the static scenario, which is expected. After applying the extended Kalman filter, the position error decreases to 1.1mm on average with a smaller standard deviation. Overall, the filter improves the accuracy with measurements that have large error, but slightly degrades the measurements with small error as well. Our experiments demonstrate that the proposed moving source localization algorithm can accomplish millimeter-level accuracy, which is sufficient for tracking the minute movements.

To evaluate the usability of vTrack in an environment with background noise, we perform experiments in various locations: an office room with general noise, an office room playing...
Electronic Dance Music (EDM), and a Starbucks cafe. General noise in an office room includes human voice and ceiling fan noise, while playing EDM adds high-frequency electronic music sound on it. We find that a cafe has all the aforementioned conditions including high-frequency machine sound, which overlaps with our audio signal and interferes the signal detection. Figure 2.15 shows the position error distribution with different background noise. Compared to the previous result performed in a quiet office room, the performance difference is marginal. Average error remains 1.3mm, 1.6mm, and 1.7mm, respectively. We confirm that our noise reduction algorithm performs well in environments with high-frequency noise.
Figure 2.15 Distribution of position estimation error with a moving sound source in different background noise scenarios.

Figure 2.16 Two additional input applications using vTrack system: (a) a larger virtual trackpad workspace with 20×26cm grid size, (b) an Apple keyboard layout printed on 18×8cm paper.

**2× trackpad:** To evaluate the impact of the size of workspace on the tracking accuracy, we expand the size of virtual trackpad to 2 times larger (i.e., 20×26cm grid) than the one used in the previous experiments. The width of the workspace exceeds the distance between microphones 1 and 2 as shown in Figure 2.16a. Average accuracy of vTrack with 2× trackpad is 1.5mm, where the degradation of its performance is negligible. The results indicate that vTrack can be used on a large surface near the receiving device, and the workspace is not
limited to the distance between two microphones.

**Keyboard input application:** To evaluate the usability of vTrack as an input interface, we design a keyboard application that tracks the sound source and recognizes the key a user selects. We use the Apple keyboard layout printed on 18×8cm paper, which is 70% of its original size. Each key is a 1cm×1cm square with 2mm spacing exists between each other. A short pause on a key generates repeated position of the key and delivers the user's intention. We perform 100 trials of experiment with a large number of words with both densely and sparsely populated keys, and achieve 95% of key recognition accuracy using vTrack.

## 2.6 Conclusion

In this chapter, we propose a virtual trackpad interface by applying the sound source localization technique in the context of mobile systems. vTrack utilizes a minimal set of low-cost sensors on the off-the-shelf mobile devices, including a speaker on the peripheral side and a multi-channel microphone array on the receiving device. Our experiments show that vTrack achieves a reasonable positioning accuracy in both static and moving sound source scenario. We believe that vTrack's positioning performance is promising enough to demonstrate the potential of acoustic-based mobile interaction interfaces.
Chapter 3

3D Motion Tracking through Sound Source Localization

In this chapter, we propose 3DTrack system that tracks a user's free-form hand movement in a 3 dimensional space using off-the-shelf mobile devices. By attaining various cues from the characteristics of sound, 3DTrack's novel sound source localization technique enables mobile devices to interact with any sound-emitting peripherals including smartphones and wearable devices. More specifically, we model the 3D workspace using the time difference of arrival values observed at the microphone array, and develop a candidate selection algorithm using the inverse distance law of the sound amplitude and the Doppler effect of sound frequency to infer the depth of the position. Our evaluations show that 3DTrack achieves high tracking accuracy with an average error of 6.8cm in 3D tracking. The tracking accuracy increases to 4.3cm when considering the performance of 2D trajectory.

3.1 Introduction

Motion tracking in a three-dimensional space has been widely studied in recent computer vision literature [17, 39, 49], which heavily relies on RGB and infrared cameras. Motion sensing peripherals for video gaming consoles including Microsoft’s Kinect [33] and Sony’s PlayStation Camera [50] both use a dual camera setup to track the movement of objects. Recently, phone-based virtual reality solutions such as Samsung Gear VR [47] and Google's Cardboard VR viewer [13] have extended the range of mobile interaction from touchscreen to 3D space by using their rear camera. However, image-based motion tracking techniques
Figure 3.1 3DTrack’s application scenario for a virtual reality (VR) device setting. A head-mounted mobile device tracks the movement of sound-emitting wearable devices such as smart watches.

are computation-intensive due to its continuous video recording and processing at a cost of power drain [27, 28]. These are a burden for mobile devices that usually have limited computing and power resources. Moreover, the most critical drawback of a camera-based approach is that it can process the user input only when the point of interest is within its view. Accordingly, the range of interaction interface is limited by the camera’s field of view.

In this chapter, we propose a 3D motion tracking solution named 3DTrack based on the sound source localization technique. By adopting the audio signal as the medium, 3DTrack leverages a multi-channel microphone array and a speaker, which are embedded on modern mobile devices by default. Figure 3.1 illustrates a target application of 3DTrack, where a user interacts with a mobile device attached to the head-mounted VR device using sound-emitting peripherals such as smartwatches and wearable devices. We assume a scenario where the observing mobile device is placed vertically, allowing the back of the device to face the 3D workspace. The workspace is not limited to the user’s angle of view but is defined by a wide-range of 3D space near a mobile device, as far as one’s hands can reach. Our solution can be used together with the camera-based tracking approach to compensate the angle limitation and also the accuracy.

3DTrack solely relies on audio-related sensors and does not employ any other hardware components such as camera, inertial motion sensors (e.g., accelerometer and gyroscope), position sensors (e.g., geomagnetic field sensor and proximity sensor), and RF transceivers. Achieving high-level accuracy with a minimum set of sensors results in many research questions. First of all, theoretical positioning resolution is determined by the audio sampling rate, while the moving sound source deteriorates the positioning accuracy due to the uncertain-
ties included in the real-world measurements. Also, finding the depth information in a 3D workspace without additional infrastructural support leaves the decision making process more challenging compared to the 2D scenario.

We address the challenges by automatically modeling the 3D workspace using theoretical values of time difference of arrival (TDoA) observed at the microphone array. We find the mapping between TDoA measurement values and the actual distance describes the target 3D space well. Then, we develop a candidate selection algorithm, which performs a modified k-nearest neighbor search from the 3D model to estimate the position of sound source. As the readings are error-prone when the sound source is moving, we investigate additional cues from amplitude and frequency change of the audio signal. Finally, we apply extended Kalman filter (EKF) to further enhance system robustness and reduce uncertainties.

We implement 3DTrack system using two off-the-shelf Android devices and demonstrate the feasibility and effectiveness of real-time 3D tracking. We show that 3DTrack can track the user trajectory with 6.8cm average error in a 3D space by comparing the results with Kinect sensor's ground truth value. Although much of the previous work allows tracking the trajectory in the air, they actually target to operate on virtual 2D surfaces. Therefore, we measure the 2D performance of 3DTrack. It shows 4.3cm tracking accuracy, which is comparable with the existing solutions.

3.2 Related Work

In this section, we summarize relevant work to 3DTrack. We focus on the literature that tracks users’ motion by adopting various methods including camera, audio signal, motion sensors, and RF signals, in the context of mobile interaction interface.

3.2.1 Vision-based Tracking

Kinect-style RGB and depth cameras are widely used in the recent computer vision literature for 3D gesture recognition [17]: from hand articulations [39] to body joints tracking [49]. However, camera-based tracking requires users to establish line-of-sight with the camera infrastructure, which has limited field of view (i.e., angle of view). 3DTrack can compensate the limitation of camera-based scheme and support a wide-range workspace outside of the camera.
3.2.2 Audio-based Tracking

By the virtue of multi-channel microphone array and a speaker embedded in modern mobile devices, acoustic ranging schemes are actively studied in the mobile systems field. Based on BeepBeep [40], a pioneer distance ranging procedure, [45] first proposes phone-to-phone static ranging in a 3D space, where both methods assume the devices to be static. Sword-Fight [64] presents an advanced computation scheme for fast-moving mobile devices. These schemes rely on the time of arrival (ToA) of the sound, which estimate the distance between time-synchronized devices by exchanging the send and receive timestamps. As our work assumes asymmetric resource for all participating devices that are not time-synchronized, we use time difference of arrival (TDoA) of the sound.

TDoA is commonly used in a navigation technique called multilateration, which requires a number of readings from multiple receivers to estimate exact position of the target. Global Positioning System (GPS) and GSM signal-based mobile phone tracking are the most representative applications using the TDoA. [30, 66] study keystroke snooping scenario using the keyboard typing sound observed at nearby mobile devices, while Toffee [58] enables finger-tapping interface around the device using four vibro-acoustic piezo-electric sensors attached underneath of each corner of the mobile device. To the best of our knowledge, 3DTrack is the first work to present a centimeter-level tracking interface that leverages the TDoA of sound in a 3D space.

From distance ranging, researchers have shown that audio signals can also be used as a medium for mobile input. For a low-granularity gesture recognition around mobile devices, frequency shift of the sound wave incurred by the in-air movements is popularly used in the literature. SoundWave [16] and AudioGest [46] decode the echo-signal using a single laptop computer, which detect simple hand gestures and their combinations. AirLink [7] further allows multiple mobile devices to share files by the hand-waving gesture. DopLink [4] and Spartacus [53] make a sound-emitting mobile device to interact with other devices by pointing at the target device.

Aiming higher granularity to track minute movements, FingerIO [35] transforms a mobile device into an active sonar system using OFDM-modulated ultrasound to enable finger input interface, while AAMouse [62] turns a mobile device into an air mouse that interacts with a smart TV equipped with two speakers using the Doppler shift. However, theses schemes are intended to operate only in 2D spaces (i.e., horizontal planes), while 3DTrack is feasible in extracting the third-axis data for full 3D tracking information.
3.2.3 Sensor-based Tracking

Other than the audio signals used in our work, inertial motion sensors are actively adopted for mobile input interface. PhonePoint Pen [2] allows the user to write in the air by holding his mobile device. Also, various wearable devices such as rings [15, 36] and smartwatches [60] are adopted to track one's writing or keyboard typing input on 2D surfaces. ArmTrak [48] achieves 3D posture tracking of the entire arm using a smartwatch as a single point of measurement, but it involves considerable latency in retrieving accurate results. Low-power sensing ability is a definite advantage of motion sensors, while they are widely known to produce inaccurate tracking results with accumulative error when solely used.

3.2.4 RF-based Tracking

Gesture recognition using RF signals has been extensively investigated for mobile interaction. SideSwipe [65] uses unmodified GSM signal that is always observed by default at mobile devices to recognize in-air gesture near the device. However, the gestures are limited to pre-defined ones with low granularity and are classified through machine learning techniques. Some recent studies employ commodity Wi-Fi cards [51], software radios [22], and RFID tags [54] to track the precise hand trajectory or handwritten text in the air, which require external devices such as laptop computers and RFID readers connected with omni-directional antennas. 3DTrack operates on unmodified mobile devices in spontaneous scenarios.

3.3 3D Motion Tracking

3DTrack system design is motivated by the human auditory system, which accurately perceives sound in 3D space based on binaural cues including interaural disparities in time and sound level. We introduce the core idea of 3DTrack system in this section.

3.3.1 3D Space Modeling using TDoA

3DTrack utilizes the previous work vTrack as a building block, which presents a millimeter-level sound source localization scheme on a 2D space. The vTrack system fingerprints the time difference of arrival (TDoA) values measured at three microphones on each point of the grid, which demands a manual DB construction procedure before the first run. However, it
is quite difficult to define a workspace and accurately collect measurements in a 3D space. Therefore, instead of utilizing the fingerprint DB, we convert the theoretical distance between the sound source and mobile device to the corresponding TDoA values and build a TDoA model which describes the 3D space near the mobile device.

To verify how well the distance derived from TDoA measurements reflect the theoretical distance, we evaluate the differentials between them. For experiment, we mimic the setup of the VR application and make a $90 \times 60 \times 60$cm rectangular cuboid with a 15cm grid on each edge. The receiving mobile device is attached at the front of cuboid, while the sound source is allowed to move inside the cuboid as shown in Figure 3.2. We first choose the Cartesian coordinate system for modeling the three-dimensional Euclidean space, where the left lower corner of the cuboid is labeled as (0, 0). Then, the TDoA values are repeatedly collected at each grid point of the cuboid.

The average of distance differential of theoretical and measured values is 3.0mm, with a standard error of 0.1mm. To study the impact of distance, azimuth angle, and altitude angle of the sound source measured from the mobile device, we borrow parameters from the auditory system as illustrated in Figure 3.3, which is similar to the spherical coordinate system. From the point of view of the mobile device, azimuth angle stands for the angle between the sound source and y-axis, which is orthogonal to its screen plane (i.e., left and right). Altitude angle denotes for the angle between the sound source and its projection on the XY plane (i.e., up and down). Distance parameter refers to the Euclidean distance between the sound source and the mobile device.
Figure 3.3 3D coordinate of a sound source is defined by altitude, azimuth angles, and distance from the receiving mobile device.

The distance differentials are as low as 1.2mm when the sound source is at one-hop distance (i.e., 15cm) from the mobile device. The differential increases as the distance between the two objects increases, but tends to converge on the average value, as shown in Figure 3.4a. When the distance exceeds 70cm, the differential shows an abrupt increase, which may lead to the ambiguity of the sound source localization at a distance. Table 3.1 is the anthropometric dimensional data provided by NASA [3], which includes the body segment dimensions such as shoulder-to-elbow, elbow-to-hand, and hand lengths. We compute shoulder-to-forearm length by subtracting the hand length from the summation of shoulder-to-elbow and elbow-to-hand length. Based on this data, we infer the operation range of a smartwatch which is worn on the wrist will fall within the 70cm range in most cases.

Table 3.1 Anthropometric dimensional data of shoulder-to-wrist length in cm.

<table>
<thead>
<tr>
<th>Gender</th>
<th>%</th>
<th>Shoulder - elbow</th>
<th>Forearm - hand</th>
<th>Hand</th>
<th>Shoulder - forearm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>5%</td>
<td>27.2</td>
<td>37.3</td>
<td>15.8</td>
<td>48.7</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>29.8</td>
<td>41.7</td>
<td>17.2</td>
<td>54.3</td>
</tr>
<tr>
<td></td>
<td>90%</td>
<td>32.4</td>
<td>44.6</td>
<td>18.7</td>
<td>58.3</td>
</tr>
<tr>
<td>Male</td>
<td>5%</td>
<td>33.7</td>
<td>45.1</td>
<td>17.9</td>
<td>60.9</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>36.6</td>
<td>48.3</td>
<td>19.3</td>
<td>65.6</td>
</tr>
<tr>
<td></td>
<td>90%</td>
<td>39.4</td>
<td>51.7</td>
<td>20.6</td>
<td>70.5</td>
</tr>
</tbody>
</table>
According to our experiments, the differentials are around 2mm or lower when the sound source is on the same plane with the receiver's screen (i.e., XZ plane in Figure 3.3). As the microphones are embedded perpendicular to the edges, sound undergoes less attenuation when the sound source is near 90° azimuth angle or 90° altitude angle as presented in Figure 3.4b and 3.4c. With other azimuth and altitude angles, the differentials are marginal and show values near average. From this result, we conclude that the TDoA values derived from the theoretical distance have minor differences compared with the actual TDoA measurements, and they describe the 3D space near the mobile device well. 3DTrack uses this TDoA...
Figure 3.5 Modeling results of 3D workspace database using theoretical time difference of arrival values computed by different pair of microphones.

database to perform the $k$-nearest neighbor search and finds the best matching tuple with the TDoA values measured in real-time.

Figure 3.5 presents the 3D databases that are constructed using the theoretical values of TDoA measured at different combination of microphones. DB in Figure 3.5a contains values derived from horizontally-located microphones (i.e., mics 1 and 2), while Figure 3.5c shows the results from microphones that are placed vertically (i.e., mics 2 and 3). The values are distinguishable from its neighbors when the distance from the sound source and mobile
Figure 3.6 Time difference of arrival measures by two microphones 1 and 2 can localize a sound source on a surface (hyperboloid 1) in 3D space. With three microphones, the position is identified on a line (shown in red), which is the intersection of two hyperboloids 1 and 2.

device is short. However, as the distance increases, the differentials decrease, as shown in the back view of the DB in Figure 3.5b. This magnifies the confusion in selecting the candidate position using the KNN matching process, especially in determining the depth information with the moving sound source.

3.3.2 Candidate Matching from DB

TDoA is noted for its merit that does not require a clock synchronization between the sound source \( S \) and its receivers \( R = [R_1, \cdots, R_N] \). Distance \( d_n \) from source \( S \) to receiver \( R_N \) cannot be directly determined from the signal reception time \( t_n \) without knowing the time sent by the source. Instead, the distance differences between receivers can be computed by using the time delay between them. When a pair of receivers, mics 1 and 2, and a TDoA value are given, the loci of possible sound source are described as one-half of a two-sheeted hyperboloid in a 3D space, where two receivers are the foci of the hyperboloid. With a third receiver present, a second TDoA measure from mics 1 and 3 leads to the second hyperboloid, which makes an intersection with the first hyperboloid as shown in Figure 3.6. Although mics 2 and 3 give a third TDoA value, it is dependent on the first and the second TDoA and fails to provide an effective information for positioning the sound source. In general, to pinpoint the exact location of the sound source in 3D, there should be at least four receivers that can produce three unrelated TDoA measures.
Let $D_{n,1}$ be the TDoA distance estimation measured with respect to the first receiver $R_1$. Then,

$$D_{n,1} = d_n - d_1 = c(t_n - t_1), (n = 2, \cdots, N)$$

(3.1)

where $c$ is the speed of sound in dry air at 20°C (343.2 m/s). Given the coordinates of receivers $R_n = (x_n, y_n, z_n)$, the source coordinate $S = (x, y, z)$ can be estimated by solving the following equations.

$$\begin{align*}
D_{2,1} &= \sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2} \\
&\quad - \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} \\
&\quad \vdots \\
D_{N,1} &= \sqrt{(x-x_N)^2 + (y-y_N)^2 + (z-z_N)^2} \\
&\quad - \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2}
\end{align*}$$

(3.2)

With three unknown variables $x, y$, and $z$, the minimum number of receivers is four, which can give at least three related equations. Therefore, we can narrow the choice to a second degree curve, but not to a unique point with three microphones, leaving the ambiguity.

We first perform KNN-search using the 3D TDoA model by placing the sound source static at each grid point. Figure 3.7 shows the best and the second-best matches returned by the KNN-search algorithm. The best match finds the exact grid index with 95% of probability, and achieves 100% positioning accuracy when complemented by the second-best match.
result. When considering the second-best candidates alone, they fall within one-hop distance of the ground truth grid in 74% of cases, and within two-hop grid by 93%. However, when the sound source moves in a 3D space, the sound propagation suffers distortion due to multi-path propagation and fading, resulting in positioning error. We find from our experiments that this error tends to be bounded by the second-best candidate results, in addition to the aforementioned possible positions on the same curve. Therefore, we introduce the difference of amplitude during the movement to determine the exact position of the sound source.

3.3.3 Handling the Moving Sound Source

The raw data generated by a microphone represents the instantaneous voltage output. It is related to changes in sound pressure from the local average. When a microphone records sound, the air pressure around the microphone goes up and down slightly, which moves a membrane back and forth producing an alternating voltage value (i.e., periodic acoustic pressure vibration). Amplitude with regard to the sound pressure $p$ decreases inversely proportional to the distance $r$ from the sound source. The relationship $p \sim 1/r$ is known as the inverse distance law of the sound.\(^1\) Based on this attribute, we can estimate the distance between the sound source and the receiver using the change of the amplitude, which can be an additional cue in selecting the correct candidate.

Assume an example scenario shown in Figure 3.8 where the sound source was observed with amplitude 0.6 dBm and distance 36 cm at timestamp $t_{i-1}$. On the next timestamp $t_i$,

\(^1\)The widely-known inverse square law also represents the sound-distance relationship but using the sound intensity $I$ (i.e., energy quantity). As $I \sim 1/r^2$ and $I \sim p^2$, thus $p \sim 1/r$. 

![Figure 3.8 Candidate selection algorithm of moving sound source scenario. Based on the inverse distance law of sound, change in sound amplitude is used to infer the distance of sound source from the receiver.](image-url)
the mobile device perceives the amplitude of the sound signal to be 1.2dBm, while the KNN search returned two candidate positions: the best match at distance 48cm and the second best at 22cm. When comparing the current observation with the previous, doubled amplitude implies the distance is dropped by half of its previous value, 18cm from the receiver. Therefore, we select the second-best candidate with distance 22cm, which is at a similar distance to the estimate in this case. However, the algorithm may end up with excessive error by accumulating distortions even when the previous observation contains minor error. To prevent this, we collect amplitude at a known grid point and calibrate the possible errors using this reference value.

Additionally, we leverage the frequency shift that results from movement of the sound source by measuring Frequency Difference of Arrival (FDoA) at the microphone array. As the sound source moves, its relative motion with respect to the receiving microphones results in different Doppler shifts observations of the sound source at each position. The displacement $D$ is computed using the base frequency $f$ and the observed frequency $f'$ with speed of sound $c$:

$$D = \left(1 - \frac{f}{f'}\right)c \cdot t_s$$

(3.3)

where $t_s$ is the time interval between two consecutive readings. The term $(1 - f/f')c$ is the velocity of the sound source, which also contains the direction information: positive when approaching the receiver and negative during the recession. These cues are concurrently used in the selection algorithm to support the decision making process.

### 3.3.4 Improving the Accuracy

In addition to the candidate position returned by our KNN algorithm, we leverage TDoA directly to compute the azimuth and altitude angular parameters illustrated in Figure 3.3. By assuming the sound signal arrives in parallel as shown in Figure 3.9a, the azimuth angle $\theta$ of the sound source is known to be the same as $\angle$Mic2 by trigonometry. Therefore, using the TDoA of microphones 1 and 2 $\Delta t_{12}$, the azimuth is calculated as

$$\theta = \arcsin\left(\frac{\Delta t_{12} \cdot c}{l_{12}}\right)$$

(3.4)

where $l_{12}$ is the known distance between microphones 1 and 2. Similarly, given the TDoA $\Delta t_{23}$ and distance $l_{23}$ between microphones 2 and 3, the altitude angle $\varphi$ shown in Figure 3.9b is
Figure 3.9 Azimuth and altitude angle of arrival computation in 3D using the time difference of arrival ($\Delta t$) measure at two microphones.

Figure 3.10 Distribution of azimuth and altitude angle estimation error in 3D.

delivered.

$$\varphi = \arcsin \left( \frac{\Delta t \cdot c}{l_{23}} \right)$$

(3.5)

To evaluate the accuracy of angle estimation using the TDoA value, we compare the result with the theoretical value, which is computed based on the ground truth. As shown in Figure 3.10, azimuth angle has 2.0° average accuracy with a 1.3° standard deviation, while the altitude angle shows 1.8° estimation error with a 2.5° standard deviation. Although the
average accuracy of two angle estimations are not significantly different, their distribution and standard deviation highly varies. This asymmetry is due to the difference of the distance between two microphones. The TDoA value cannot exceed the distance between two microphones by the triangle inequality, theoretically limiting the number of possible hyperboloids to \( \lceil 2l \cdot f_s/c \rceil \) \[30\]. Therefore, there exists 175 horizontal and 37 vertical hyperboloids, respectively, given distances between microphones \( l_{12} = 15.6\text{cm} \), \( l_{23} = 3.3\text{cm} \), and audio sampling rate \( f_s = 192\text{kHz} \) in our experiments. This leads to distinct resolutions that the system can distinguish horizontally and vertically, showing different performance in the estimation accuracy.

Using these parameters, we apply extended Kalman filter (EKF) to reduce fluctuations and errors in the raw data readings. The filter uses a series of measurements observed over time, which contains noise and uncertainties. It estimates the unknown variables using Bayesian inference and a joint probability distribution of the variables. The filter produces estimates of the current state vector \( x_k = [p_x \: v_x \: p_y \: v_y \: p_z \: v_z]^\top \) during the prediction stage, which contains three-axes coordinates and velocities of the sound source. When the next measurement vector \( z_k = [r \: \theta \: \varphi]^\top \) is observed where \( r \) is the distance from the sound source to the receiver, the estimates are updated using a weighted average during the update stage: it gives more weight to the estimates with higher certainty. The advantage of the algorithm is that it can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix. We follow the details of vTrack for the fine tuning process of EKF.

3.4 Implementation

We implement 3DTrack using the vTrack system architecture and implementation. In this section, we describe differentials that are made for 3DTrack realization.

3.4.1 3D Rendering

At the receiving device, the trace of the sound source is rendered on screen for the visualization purpose. For 3D rendering, we adopt a 3D perspective projection to map the three-dimensional points to a two-dimensional plane, which reflects the perspective of the observer. The perspective projection shows distant objects as smaller to provide additional realism. A 3D point \( P = (x_p, y_p, z_p) \) behind the screen will be projected to a 2D coordinate
Figure 3.11 3D perspective projection for 3D rendering on a 2D screen. Sound source P in 3D workspace is projected onto 2D surface S (S’) from the observer’s eye position E in 3D.

\[ S = (x_s, y_s) \]

on the screen, which depends on the observer’s eye position \( E = (x_e, y_e, z_e) \) in front of the screen, Figure 3.11a illustrates the top view of the scenario. Because \( \triangle ECS \) and \( \triangle EQP \) are similar triangles, we use the property that their corresponding sides are proportional and get the following equation.

\[
\frac{EC}{EQ} = \frac{CS}{QP} \quad (3.6)
\]

By assuming the screen is at position \( y_s = 0 \) and substituting with respective coordinates, we get:

\[
\frac{y_e}{y_e + y_p} = \frac{x_s - x_e}{x_p - x_e} \quad (3.7)
\]

where \( y_e \) stands for the distance from eye to the center of the screen. Thus, the 2D coordinate \( x_s \) is represented as follows according to the coordinate naming convention used in this chapter. Also, \( y_s \) can be derived in the same fashion from the side view as shown in Figure 3.11b.

\[
x_s = \frac{y_e(x_p - x_e)}{y_e + y_p} + x_e \\
y_s = \frac{y_e(z_p - z_e)}{y_e + y_p} + z_e \quad (3.8)
\]
3.5 Performance Evaluation

We evaluate the performance of 3DTrack by letting a user hold a sound source and draw various shapes and gestures in a 3 dimensional space. Placing the receiver in front of the user, the workspace is defined by the space behind the mobile device. For all experiments, we use Samsung Galaxy Note 4 as a receiver, which allows concurrent three-channel microphone input without any hardware modification. Samsung Galaxy S II operates as the audio signal transmitter. To improve the mobility, we redirect the output wire of the embedded speaker to an external speaker.

We conduct two-phase evaluation to the compare tracking results of 3DTrack with the ground truth trajectory: use the cuboid grid testbed for micro-benchmarks and then leverage Microsoft Kinect sensor to perform macro-benchmarks. We compute point-to-point Euclidean distance from the trajectory points to the ground truth entries, where the shortest distance is regarded as the positioning error.
3.5.1 Micro-benchmarks

3.5.1.1 Trajectory Accuracy

The coordinates of 3D space is configured the same as before, where the receiver is attached at (45, 0, 30) position as shown in Figure 3.2. Figure 3.12a shows a number of parabolas crossing the 3D space in the perpendicular directions, which distinctly reflects the depth changes. Similarly, Figure 3.12b illustrates two gestures performed with different altitudes and angles, which clearly captures the minute distinctions in the movement. Figure 3.13 shows the CDF of positioning error in the 3D cuboid space: it has 6.6cm accuracy with a 3.3cm standard deviation.

3.5.1.2 Impact of Extended Kalman Filter

Figure 3.14 shows two trajectory results: raw data returned by the search algorithm, and data after applying extended Kalman Filter. The raw data may have outliers incurred by the uncertainties of sound propagation during the movement. Extended Kalman filter reduces errors and smooths the trajectory based on the previous input and estimates the current position. The raw data before applying the EKF has 7.1cm of position error with a 3.6cm standard deviation as illustrated in Figure 3.13. By adopting the EKF, we gain 7.0% of increment in the positioning accuracy.
Figure 3.14 Raw tracking data and calibrated tracking result by applying extended Kalman filter.

(a) In a full 3D space

(b) On a 2D plane

Figure 3.15 Tracking results performed on a virtual surface in three-dimensional space and their projection results on a two-dimensional plane.

3.5.1.3 2D Performance

The error increases in 3D compared to the 2D scenario because of the third dimension added for computation. Most of the recent work in the literature measures its 2D performance even it operates in 3D. Therefore, we also measure the 2D performance of 3DTrack by projecting the trajectory on a plane in the 3D space for comparison purposes. Figure 3.15a presents triangle shapes drawn in 3D but targeted on the XZ plane, where the ground truth path has a
constant depth (i.e., unchanging y value). Figure 3.15b illustrates the same traces on the XZ plane. We compute the accuracy of trajectories by leaving the y-axis value out to evaluate the 2D tracking performance. As shown in Figure 3.13, 3DTrack has 4.1cm accuracy with a 3.1cm standard deviation when projected on a 2D plane.

3.5.1.4 Impact of Relative Position

We now evaluate the impact of the sound source position relative to the receiver on the tracking accuracy of 3DTrack. With the receiver placed at height 30cm, Figure 3.16c shows
that movements at different height (i.e., z-axis) have negligible effect on the tracking accuracy. The median position error for trajectories at lower (15cm), equal (30cm), and higher (45cm) levels are 5.9cm, 5.9cm, and 6.6cm, respectively. However, the error increases as the depth (i.e., y-axis) from the receiver increase as presented in Figure 3.16b. The median error for depth 15cm, 30cm, and 45cm are 5.6cm, 6.3cm, and 7.5cm.

Also, by defining the space within the receiver's length as the center portion, we partition the x-axis of workspace into three: left, center, and right. According to the result in Figure 3.16a, the median tracking errors are 6.3cm, 5.8cm, and 7.6cm, respectively. The movements conducted in front of the receiver have higher accuracy, while right side of the receiver has poorer performance compared to the left side. This is due to the disparity in performance of microphones: as the microphone at left side (i.e., mic 1) is inferior to other microphones on the right side of the receiver, it occasionally fails to capture the weak audio signal that comes from the right portion of the device.

3.5.1.5 Depth Estimation Accuracy

We evaluate the depth estimation accuracy of 3DTrack by comparing the estimated amount of depth displacement with the ground truth change. The estimation error is calculated as (Estimated depth change – ground truth)/ground truth, where the ground truth depth varies between 15cm and 50cm. Figure 3.17 shows that both underestimation and overestimation of depth displacement occur with similar probability. Overall, 3DTrack can capture the depth
change of 3D movements with 11.9% of average estimation error.

3.5.1.6 Distance Estimation Accuracy

To evaluate the estimation error of the trajectory distance compared to the actual distance traveled, we measure the percentage of the distance differentials to the ground truth: 
\[
\frac{\text{Estimated trajectory distance} - \text{ground truth}}{\text{ground truth}}
\]
This distance differential indicates redundant movements occurred by the position estimation error during the movement. Start and end points of a gesture are indicated by a short pause during the movement. The average of positioning accuracy for starting and ending points is 6.3cm. Considering the distance for a full trajectory of the movement, the average distance error is 11.4% of the actual distance traveled as shown in Figure 3.18. However, when we only take the distance between start and end position of the movement into account and compare it with the shortest path distance, the distance differential becomes 6.2%. Results in Figure 3.18 implies that 3DTrack can measure a line distance between two points with less than 10% of estimation error for over 80% of cases.

3.5.1.7 Impact of Operation Time

Finally, we evaluate the tracking accuracy with different duration of the operation time. We start the 3DTrack system and perform continuous sound source tracking up to 5 minutes. As
shown in Table 3.2, the median tracking error remains stable with only slight variations. We observe that 3DTrack's error does not accumulate over time, which allows substantially long tracking time with consistent accuracy.

### 3.5.2 Macro-benchmarks

To better understand the application-level performance of 3DTrack, we adopt Microsoft Kinect version 2.0 sensor for ground truth collection. Kinect is an input peripheral for both the Xbox video game console and a PC, which captures the user motion in three dimensions. The depth information is retrieved from an infrared camera embedded in addition to a RGB camera, and is streamed as a 640 × 480 pixels video at a frame rate up to 30Hz with 11-bit depth data, providing 2,048 levels of granularity. Using the SDK provided for Windows PC, we build an application in C# to collect the movement of a user's hand from skeletal tracking feature. Reverse engineering has revealed that the error of depth measurement increases as the distance from the Kinect sensor increase, and reaches 4cm at the maximum operation range of 5 meters. Thus, the operation is recommended between 1 to 3 meters from the sensor.

As the tracking accuracy of Kinect is sufficiently high in a 3D space, we regard the Kinect raw data as a reliable ground truth of user movement and compare the path difference with our scheme. Kinect sensor is connected to a ASUS laptop with Intel i7 CPU and 4GB RAM for data collection. We place the receiving Android device on a tripod standing in front of the Kinect sensor at 1.5 m distance and simultaneously collect tracing data from both Android and Kinect. Figure 3.19a and 3.19b illustrate several results, where the dotted lines represent the corresponding traces from the Kinect sensor.

Figure 3.20 shows the summary of the comparison: the position difference between 3DTrack and Kinect dataset. In a 3D space, 3DTrack shows 6.8cm difference from the Kinect trace with a 3.2cm standard deviation, which is a similar result to the previous testbed experiments. For comparison on the 2D plane, we project the 3D measurement data on the
(a) Two parabolas with different directions  
(b) Two swings with different altitudes

Figure 3.19 3DTrack’s tracking results compared with Kinect motion sensor.

Figure 3.20 Distribution of position estimation difference between 3DTrack and Kinect motion sensor.

plane by leaving the third dimension value out, and compute the Euclidean distance. The tracking average accuracy on a 2D plane is 4.3cm with a smaller standard deviation of 2.8cm. The experiment result implies that 3DTrack gives reasonable tracking accuracy for both 3D and 2D movements.
Table 3.3 Energy consumption measured during one-hour application run. Case 1 keeps only the screen on. Case 2 turns both screen and audio recording function on. Finally, case 3 activates all functionalities including coordinate computation, audio signal processing and filtering.

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
<td>⃝ 958mWh (7.7%)</td>
<td>⃝ 931mWh (7.5%)</td>
<td>⃝ 927mWh (7.5%)</td>
</tr>
<tr>
<td>Audio</td>
<td>× 0mWh (0.0%)</td>
<td>⃝ 127mWh (1.0%)</td>
<td>⃝ 546mWh (4.4%)</td>
</tr>
<tr>
<td>Computation</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

### 3.5.3 Energy Consumption

To evaluate the energy consumed by each procedure, we measure the battery level drop during the application run. However, Lithium-Ion (Li-Ion) battery’s discharge curve is not strictly linear: it shows the maximum voltage excursion from the nominal value when fully charged and the minimum voltage excursion when nearing the end of discharge voltage (EODV) point. Otherwise, it has an approximately flat discharge curve near the nominal voltage of the cell during discharge. Therefore, we perform one hour-long experiment when the battery is charged about 85% of its capacity.

We use the OS-provided native application as well as a popular application for Android [14] to measure the battery consumption, which both returned similar values with negligible differences. However, as these methods only show per-application measures with screen separately, we perform experiments for three cases: only turning the screen on, turning both screen and audio recording on, and performing all computations including the screen turned on. Total battery capacity for Samsung Galaxy Note4 is 3,220 mAh, and we convert the differentials of milliampere-hour to milliwatt-hour by multiplying the voltage of the power source, which is 3.85 V according to the device specification. The numbers in parenthesis in Table 3.3 represent the percentage of energy consumption to the battery capacity.

As expected, the screen drains most of the energy during the one-hour application run, which is over 7% of total battery capacity. However, continuous audio recording and signal processing procedure only consume 1% of the energy, as shown in the second case. This presents that always-on audio detection demands minor resources compared to the screen output. The last case denotes the burden imposed for computing the positioning algorithm such as KNN-search and extended Kalman filter. Computation requires $3.3 \times$ more energy.
than the audio processing, and consumes 3.4% of total battery when continuously run for one hour. From the experiment results, we extrapolate the amount of energy consumption for 10-minute will be less than 2% of battery capacity. We conclude that our audio-based tracking system is lightweight and can be performed with low energy on mobile devices.

3.6 Conclusion

In this chapter, we propose 3DTrack system that tracks a user’s free-form hand movement in a 3 dimensional space using the off-the-shelf mobile devices. By attaining various cues from the characteristics of sound, 3DTrack’s novel sound source localization algorithm enables mobile devices to interact with any sound-emitting peripherals such as smartphones and wearable devices. Our experiments show that 3DTrack achieves high tracking accuracy with an average error of 6.8cm in 3D tracking. The tracking accuracy increases to 4.3cm when considering the performance of 2D trajectory.
An Unobtrusive Interaction Interface for Multiple Co-located Devices

In this chapter, we propose a practical interaction interface for a local set of devices to connect to each other without requiring a lot of effort from their users. We find that highly time-correlated Wi-Fi signal measures experience less variation in their received signal strength (RSS), which alleviates the impediment of the RSS-based solution. We also notice that each room observes significantly discrete signal signature even when they are adjacent to each other. We leverage these observations and rely on the RSS measure to detect and unobtrusively pair devices in the same space. To achieve our goal, we devise a new similarity metric for RSS comparison and propose a means of selecting the cluster threshold. We implement a prototype named Flock for Android devices. By relying on our client-server system architecture, mobile users can interface with others in a few seconds without having to search-and-select nearby devices.

4.1 Introduction

Aided by radio-connected, sensor-rich smartphones, tablet PCs, and wearable devices, ubiquitous computing has arrived. Experts predict that the number of connected devices managed by one person will reach 4.3 by 2020 [38]. As the density of mobile devices increases, frequent connectivity between adjacent devices is inevitable. Marketers of mobile devices have invested much effort in designing a hands-down pairing method between devices, which aims to simplify certain aspects of the connection setup task. However, we conclude
that contemporary device pairing interfaces fail to satisfy their users.

Bluetooth and NFC are popular short-range communication technologies that utilize committed channels. Although most mobile devices are equipped with these communication hardware, they do not fully support interoperability between heterogeneous devices. For example, file sharing over Bluetooth or NFC using devices with different operating systems (e.g., iPhone and Android devices) is not possible due to proprietary restrictions from the vendors. Bluetooth and NFC cannot setup concurrent connections between more than two devices, but are limited to one-to-one pairing. Moreover, delicate physical conditions are needed to pair two devices due to their limited discovery range.

Most device pairing solutions involve time-consuming steps of searching and selecting the receiver, even when the devices are in close proximity. Considerable time and effort are needed to connect devices. The extra effort undermines the convenience of the service, especially when used by ubiquitous devices that require frequent interaction among multiple devices. For a better user experience, devices should connect in a sensory and nonintrusive manner. Most desirable is a high degree of automated provisioning towards zero configuration networking that creates a network of devices without any manual user intervention or special knowledge of the environment.

To clarify our goal, we describe two target scenarios. Alice is about to give a lecture to a class. Shortly before the class starts, she finds an interesting article that she wants to share with her students. She needs a simple way to copy the article from her mobile device and paste it to the students’ devices in the same classroom during the lecture. In the other scenario, Bob wants to retrieve a grocery shopping list from the smart refrigerator in the kitchen to his smartphone before going shopping. After searching for a grocery store on his smartphone, he needs to copy and paste the address of the grocery store from his smartphone to the smart GPS device embedded in his car.

In this chapter, we design an unobtrusive interaction interface for multiple mobile devices in proximity. We suggest an easy way for a local set of devices to connect to each other without undue effort from their users. We concentrate on a practical and low-cost approach that works robust even in the environments without any knowledge in advance. Inspired by the prospect of high-density ubiquitous devices that are equipped with a Wi-Fi communication module, we leverage the signal information observed at each device to determine the co-existence of devices. Although GPS provides accurate coordinates outdoors, its performance degrades significantly indoors. As the GPS module is notorious for its high energy consumption, we
refrain from relying on GPS in favor of signal information that can be retrieved with minimal cost.

In our system, a cloud server collects a series of Wi-Fi scan information from each mobile device. It computes the similarity of Wi-Fi RSS from different devices using a new similarity metric that infers the co-existence of mobile devices. By using the similarity feature as an identifier, devices in proximity can recognize each other. Accordingly, users are able to pair with others without searching for and specifying the receivers to establish a connection. The merits of our scheme are its wide applicability and scalability, accomplished by leveraging the signal information accessible with low cost on commodity mobile devices. The fact that it is free from labor-intensive site surveying makes it even more attractive.

By conducting experiments on a variety of sites, we demonstrate that our system can be used to determine the co-existence of mobile devices that are located within a physically separated space with high accuracy. It takes as little as two seconds for a device to discover and connect to its peer, without the necessity of leveraging any pre-arranged environmental settings. As a proof of concept, we implement a prototype named Flock for Android devices, and publish the app on Google Play Market[11]. It is a mobile communication app that automatically identifies nearby devices and pairs them to flock together in real deployment scenarios.

The rest of this chapter is organized as follows. In Related Work section, we introduce relevant issues and summarize the previous work. We then present our co-existence detection technique and verify its feasibility. System Implementation section introduces our system architecture and implementation details. Performance of the system is extensively evaluated in the Evaluation section. Finally, we conclude the chapter in the last section.

4.2 Related Work

Various techniques have been proposed in the literature to provide intuitive pairing of mobile devices that can facilitate spontaneous interactions.

Synchronized gesture detection matches accelerometer readings generated by the actions. Smart-Its-Friends [20] and Shake Well Before Use [32] sense synchronized device shaking, while Synchronous Gestures [19] detects bumping of two devices. The pointing gesture is combined with acoustic signal processing techniques. Point&Connect [41] leverages time-of-arrival of the signal emitted from the target device, while Spartacus [53] relies on the
Doppler shift effect. These strategies require users’ synchronized performance of discrete actions based on their agreement on pairing. In contrast, our work does not require hard time synchronization between devices, and it works with more than two devices.

NearMe Wireless Proximity Server [25] is most relevant to our work that leverages Wi-Fi RSS. NearMe server extracts the similarity feature using Spearman rank-order correlation, and empirically estimates the physical distance between two devices by fitting the measurement data into a distance function. Like NearMe server, PeopleTones [26] uses RSS but for GSM readings. PeopleTones simply computes a coarse-grained proximity ratio of the overlapping signals. Aforementioned approaches require labor-intensive training to obtain a distance function whenever they are applied on a new venue, and therefore lack scalability for practical deployment. In contrast, our work is universally applicable without any site survey; we rely on a more elaborate similarity metric, which takes the Wi-Fi signal strength into account.

Ensemble [23] determines co-located devices by leveraging the fact that RSS variations of transmitters in close proximity are highly correlated. The witness devices monitor RSS from the pairing devices and ascertain whether their RSS variations are correlated. However, Ensemble works best when the pairing devices are close enough to each other (e.g., 10 cm). It also requires at least three witness devices to make a reliable decision, which combine their observations to improve the accuracy. Our work has an advantage over Ensemble in that it is not restricted by the number of participants, and it does not require assistance from nearby mobile devices.

Our approach circumvents the drawbacks of short-range communication techniques: it establishes a many-to-many communication that concurrently connects a local set of devices; its interaction range is automatically adjustable to fit physically separated spaces. It supports heterogeneous devices as long as they have a Wi-Fi networking module; it provides a higher data rate than other short-range wireless communication modules, as it relies solely on Wi-Fi.

4.3 Coexistence Detection

Our goal is to determine whether a set of devices is co-located in a physically separated space. To achieve this objective, we focus on the assumption that nearby devices will observe highly similar Wi-Fi scan results, as they are covered by the same or overlapping access points. We show that this degree of similarity can be properly used in approximating physical layout of
4.3.1 Similarity Feature Extraction

To quantify the similarity of Wi-Fi signal strength, we devise a new similarity metric $C'MH$ based on Morisita-Horn index ($C_{MH}$) [21] borrowed from the ecological literature. M-H index is a widely used method for comparing species biological diversity of two or more samples. As it incorporates information on the relative abundance of species rather than their incidence (i.e., presence or absence), we find that M-H index applied to our Wi-Fi RSS scenario can reflect delicate differences in RSS, returning a more precise similarity measurement. M-H index is sensitive to changes in the abundance of the more common species [57], which is analogous to the stronger RSS from specific APs in our case.

Our modified M-H index $C'MH$ is represented as Equation (1), where two sample Wi-Fi scans $x = (x_1, x_2, \ldots, x_{S_x})$ and $y = (y_1, y_2, \ldots, y_{S_y})$ are given. It computes correlation between two samples and is normalized to return a measure between 0 and 1 with the maximum value obtained by two identical samples [6].

$$C'MH = \frac{2 \sum_{i=1}^{S_{xy}} x_i y_i}{\left( \frac{\sum_{i=1}^{S_x} x_i^2}{x^2} + \frac{\sum_{i=1}^{S_y} y_i^2}{y^2} \right) XY}$$  \hspace{1cm} (4.1)

where:

$x_i = RSSI_{norm}$ of $AP_i$ in the scan result $x$

$y_i = RSSI_{norm}$ of $AP_i$ in the scan result $y$

$X = \text{Sum of } x_i$ for all APs in the scan result $x$

$Y = \text{Sum of } y_i$ for all APs in the scan result $y$

$S_x = \text{Number of APs in the scan result } x$

$S_y = \text{Number of APs in the scan result } y$

$S_{xy} = \text{Total number of APs in both scan results } x$ and $y$

Minimum value of Received Signal Strength Indicator (RSSI) depends on the module vendor, while the maximum value is usually considered as 0. Android developer document also describes that the range of return value of $\text{getRssi()}$ API is undefined and is not
normalized [56]. However, MIN_RSSI value is defined as -100 by Android framework source code (WifiManager.java). According to the code, anything smaller than or equal to this value are considered as intermittent to no operation. Thus, we assume that RSSI value retrieved from getRssi() function may fall between 0 and -100. We normalize the RSSI value to have a value between 0 and 1 by Equation (2) and use it in the $C_{MH}'$ index computation.

$$RSSI_{norm} = 1 - \frac{|RSSI|}{100} \tag{4.2}$$

4.3.2 Similarity Metric Verification

To verify our similarity metric, we conduct experiments in an engineering building on our campus. The measurement setup consists of one corridor and three classrooms that are divided by cement walls. Wi-Fi scan results are collected at every point shown in Figure 4.1 using a Samsung Galaxy S2. We choose four different orders for data collection to reflect various orientation factors during the measurement: sequential increment, odd-even increment, sequential decrement, and odd-even decrement order of the label number. We also perform each experiment at a different time of day to apply time variance in the dataset. Our scheme does not require any details of the experimental environment such as the location of APs,

Figure 4.1 Experiment setup in an engineering building. 116 measurement points (corridor: labels 1~56, classroom 1: labels 57~72, classroom 2: labels 73~88, classroom 3: labels 89~116) are spaced 2.4 meters apart. Each color maps a color-labeled AP and its signal coverage, while black-labeled APs are targeted to cover each classroom they are located.
which allows easy and wide deployment in real world scenario.

First, we compare our metric with Spearman rank-order correlation coefficient ($\rho_s$), which is one of the similarity features used in the prior work [25]. The coefficient value ranges from -1 to 1: -1 indicates negative correlation and 1 means positive (i.e., exact) correlation between rankings, while 0 stands for no correlation. The authors of [25] state that considering the rank of APs according to their RSS rather than the exact value is more robust to signal fluctuation, because the ranking of the APs will remain the same. However, we find that this metric is highly susceptible to unexpected results when two RSS measures are negatively correlated. In this case, $\rho_s$ value cannot be used alone as a metric for inferring the signal similarity. On the other hand, our metric seems to have a high correlation with the physical distance.

Figure 4.2 shows an experiment result that well-describes the aforesaid phenomenon. It compares the two metrics when applied to the same dataset, and presents the similarity measured between the measurement point 1 and all other points in the corridor (up to label 56 in Figure 4.1). Our similarity value $C_{MH}'$ decreases almost linearly with increasing measurement label number, where actual distance also increases. However, $\rho_s$ value shows sharp variation along the corridor, disregarding the physical layout of measurement labels. Whenever the rank order of observed APs changes, $\rho_s$ returns a negative value even when the measurements take place in proximity. A slight fluctuation in Wi-Fi signal pattern may result in weak consistency with real disposition.
Next, we compare the similarity index computed between each point in the corridor with the ground truth distance as shown in Figure 4.3a and 4.3b, and measure the correlation between two datasets. To match the ground truth distance (i.e., dissimilarity), our similarity measure is simply converted to represent dissimilarity \((1 - C'_{MH})\), where low dissimilarity stands for high similarity. Similar to the ground truth value, \(C'_{MH}\) index shows high similarity for the adjacent points, while the similarity gradually decreases when actual distance increases. Interestingly, similarity values between measurement labels around 40 up to 56 are almost identical in Figure 4.3b. Because of the large coverage of a group of APs located

![Figure 4.3 Heatmaps of (a) the physical distance and (b) dissimilarity measures between all measurement points in the corridor.](image)

Table 4.1 Correlation coefficients between the ground truth and multiple dataset using different distance metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhattan distance [37]</td>
<td>0.3410</td>
</tr>
<tr>
<td>Proximity ratio [26]</td>
<td>0.6308</td>
</tr>
<tr>
<td>Euclidean distance [29, 37]</td>
<td>0.6564</td>
</tr>
<tr>
<td><strong>Proposed metric</strong></td>
<td><strong>0.8789</strong></td>
</tr>
</tbody>
</table>
in classroom 3, the measurement points that adjoin to this room have very similar signal traits. Although such large signal coverage may obfuscate the sharpness of our similarity index, the correlation coefficient between physical distance measure and $C'_{MH}$ index is 0.8789, indicating a strong positive relationship between two datasets.

Similarly, we apply various distance metrics including Manhattan distance, Euclidean distance, and proximity ratio that are used in the literature to the same dataset for comparison. Proximity ratio [26] is a simple ratio of the number of common APs between two readings and the average number of APs observed in both readings. The terms similarity (i.e., overlap) and dissimilarity (i.e., diversity or distance) are used interchangeably in the ecological context, thus distance index is also applicable to the similarity index [6]. We compute the correlation coefficient with the ground truth value and summarize the result in Table 4.1. Our proposed metric presents the highest correlation value among others, which properly reflects the physical layout of devices by solely leveraging the RSS information.

From the same experiment environment shown in Figure 4.1, we now compute the similarity measure for each classroom (label 57~116) and the adjacent part of the corridor (label 31~56). As illustrated in Figure 4.4, each room in addition to the corridor has similar signal trait inside, which is also perspicuously discrete to that of other spaces despite their juxtaposition. From this result, we conclude that the similarity index $C'_{MH}$ alone applied to the Wi-Fi RSS patterns returns a reliable guideline in inferring the coexistence of devices in the same room.
4.3.3 Impact of Signal Fluctuation

Although Wi-Fi signal is well known for its fluctuating nature due to the multipath effect of radio signal propagation, we find that the probability of observing less fluctuation among different RSS measures increases when those measures take place in a short period of time. In other words, we can expect some benefit from the concurrency of signal observation. To verify this attribute, we perform rolling-window analysis on a time series of RSS readings. Standard deviation for each window is computed (i.e., moving standard deviation) to measure the signal variability of the sample. We vary the time window and test the influence of window length on the signal deviation. Additionally, we subsample from each window and make the number of samples equivalent for every window length.

Hour-long RSS readings are collected from nearby APs at a static position with the RSS sampling interval set to one second. Figure 4.5 depicts cumulative density function (CDF) of standard deviation of RSS applying three different window lengths. According to the experiment result, when RSS measures occur concurrently or at short intervals (e.g., within one minute), the standard deviation of RSS is less than 1.5 dBm in 80% of the cases. As the measurement interval increases, so does the probability of observing more deviation. The graph gradually converges to the 10-minute window curve in the CDF. As we target a scenario where RSS measures take place simultaneously in a highly time-correlated manner, we believe that the influence of RSS variation can be alleviated by leveraging this observation.

To validate the effect of this concurrency on our metric, we compute the similarity $C'_{MH}$
between two Wi-Fi scan readings within a rolling-window frame, using various window sizes. 
As shown in Figure 4.6, RSS readings observed within one-minute interval are more likely to 
give greater similarity value when compared with the results that are measured with longer 
window frames. For example, the probability of two scans to have similarity greater than 0.94 
is 45% when measured within one minute. This probability sharply decreases to 20% when 
the time interval increases to 5 minutes. The result indicates that Wi-Fi RSS readings that 
occur simultaneously or within a short time frame are likely to have high similarity despite 
the signal variation. This finding is suitable for our detection of physically co-existing devices.

Similarly, cumulative scan results can mitigate the influence of signal fluctuation. Instead 
of relying on a single scan, merging multiple consecutive scan results may better describe 
the signal trait of a specific space. How should we choose the representative RSS value from 
multiple scans, and how well will it reflect the actual signal attributes? The number of scan 
results to utilize may come into question, because retrieving Wi-Fi scans consumes both the 
user's time and the device's energy. To answer these questions, we first derive the average RSS 
of each AP observed within one minute frame using the rolling window approach, and set it 
as the reference value that delineates the RSS of each time frame. Next, we merge various 
number of successive scans into a single result by modifying the number from two to five. As 
the strategy to choose the representative value (e.g., minimum, maximum, and average of 
multiple scans) has negligible impact on the similarity measure, we state only the experiment 
result using the average value of RSS of each observed AP. Finally, we compute the similarity
measure $C'_{MH}$ between the reference RSS and cumulative scan results.

Figure 4.7 shows complementary CDF (CCDF) of the similarity measures for a single Wi-Fi scan and diverse number of cumulative scan results. As the number of scans merged into the representative increases, the probability of observing high similarity with the reference RSS value increases in sequence. However, the gain achieved by additional scans gradually decreases. For instance, the probability of a Wi-Fi scan having a similarity greater than 0.96 nearly doubles when merging two sequential scans compared to a single scan, but this benefit is reduced by 2.4, 2.6, and 2.7 with additional scans. Based on our experiment result, we conclude that the average value of two cumulative scan results is the optimal choice in reducing the effect of signal fluctuation.

4.3.4 Impact of the Number of APs

We analyze the impact of the number of overlapping APs and their RSS to the similarity measure. For two example scan results $x = (x_1, x_2, \ldots, x_{15})$ and $y = (y_1, y_2, \ldots, y_{18})$ where the number of overlapping APs is 14, we simulate changes in the similarity measure by controlling the overlaps. The similarity index of the original data is 0.8848. Starting from zero APs in common, we sequentially add overlapping APs in two diverse orders based on the APs’ RSS: increasing order where weak signal AP comes first; and decreasing order where strong signal AP comes first.
In Figure 4.8, bar graphs represent the similarity index, while plots denote the corresponding average RSS of the AP in log scale. The result illustrates that strong signal APs result in high similarity, and the presence of multiple APs intensifies this tendency. Although weak signal APs contribute little to the similarity level by itself, a series of APs does help increasing the similarity. Specifically, three strong signal APs achieve notable similarity level. Weak-signal APs also play a role in gradually increasing the similarity, although the signal strengths are considerably weak (e.g., near -90 dBm).

### 4.3.5 Co-located Device Discovery

Based on the similarity of the signal information, devices need to be grouped with others in the vicinity. As our goal is to propose an unobtrusive detection technique that does not involve any supervised learning task, the problem should be solved with unsupervised procedure such as clustering. However, we cannot use the well-known $k$-means clustering because it is impossible to determine the number of clusters in real time. Instead, we adopt hierarchical clustering, a method that investigates grouping of a dataset simultaneously over a variety of scales by creating a cluster tree. In this tree, leaves represent measurement points and the length of the paths between leaves stands for the distances between points. During the clustering process, distance between two clusters is determined by the farthest distance between any pair of their elements. This complete-neighbor linkage method is known to find compact clusters from the dataset, which fits with our goal.
There remains, however, another problem of where to cut the tree in order to determine the number of clusters. This is an open issue in statistical learning, and usually incurs human intervention for the decision-making. To determine a proper threshold value which can be automatically adjusted according to the deployment scenario, we perform hierarchical clustering on 100 sessions of data collected from 10 venues. Each place is composed of two adjacent rooms or stores that are physically divided by a cement wall. Based on our empirical analysis, we find that a link which has the highest standard deviation of all the links included in the calculation provides an agreeable hint to cut the tree. Therefore, we choose the average of the link heights under the maximum standard deviation link as the threshold value, and prove the generality of this strategy as follows.

Figure 4.9 illustrates the ratio of the corresponding clustering results to the total number of experiment trials. Result with two clusters represent the measurement points in two individual spaces are accurately divided according to their physical location, which is an ideal situation. Three clusters case indicates that all measurement points in one room are properly grouped into a cluster; those in the next room are split into two sub-clusters because the links are above the threshold value. Remaining instance denotes the measurement points are divided into more than four clusters, without any correct clusters on either side of the room.

Location labels 1 and 2 are classroom environments similar to Figure 4.1. As most classrooms have dedicated APs and are delicately configured for uniform Wi-Fi signal coverage,
each classroom has its own signal trait. Therefore, the proposed cluster threshold clearly divides the measurement points with high accuracy. Labels 3 to 6 include measurements collected from neighboring stores such as cafe, bakery, and restaurant. Although the stores provide their own Wi-Fi connections, the deployment is neither planned nor agreed upon by the business owners but managed individually. Due to the overlaps in signal coverage, the percentage of clear distinction of two adjacent stores remains about 90% of the experiments. The last group, labels 7 to 10 involves rooms in an apartment complex, which has a single AP that covers entire area of the house. This is the most demanding scenario because the rooms are not large enough to have distinctive signal traits. Consequently, the success ratio to get a clear-cut cluster becomes lower, as shown in Figure 4.9.

For the last case, we can opportunistically take Bluetooth signal information into account to improve the clustering performance. We adopt Wi-Fi signal instead of the Bluetooth signal because it provides consistent signal coverage over a wide range, which can be utilized as general information about the environment. In contrast, Bluetooth modules that are most commonly used in mobile devices have a short range (i.e., 10 meters or 33 feet), and are easily turned on and off by users resulting in a sporadic signal trait. However, we can use this attribute as a filter to refine the clustering result when used in small-room scenarios. According to the experiment result, if there is at least one discoverable Bluetooth device in each room, 67% of erroneous instances with three or more clusters can be adjusted to two correct clusters.

Our observation shows that the proposed threshold selection method can accurately cluster and consequently discover co-located devices in general usage scenarios.

### 4.4 System Implementation

We use our co-located device detection technique to build a proximity-based communication system. Although our initial implementation targets Android devices, note that the prototype can be applied to any commodity mobile devices and consumer electronics provided with an open API for Wi-Fi networking module. We present a practical messaging application Flock, which works in real deployment scenarios with no need of extensive site survey or infrastructural support. It automatically identifies nearby devices and immediately establishes connection with ease. Mobile users are able to flock without explicitly searching for and specifying the receiver's identifier.
Figure 4.10 Server collects Wi-Fi signature from clients and clusters them by computing similarity of signatures. Then, the server connects clients in the cluster using messaging protocol. Clients perform Wi-Fi scan and communicate over the network.

The design of our system is twofold: the mobile client handles device positioning, networking, message authentication, and user interaction, while the cloud server manages data collection, peer device discovery, Wi-Fi scan result and message database, and networking with clients. Figure 4.10 shows the system architecture of our application.

4.4.1 In-phone Processing

4.4.1.1 Positioning Unit

When a user starts the client, the positioning unit collects the Wi-Fi scan result using WiFi-Manager API. We record BSSID which is a unique MAC address of the AP instead of Service Set IDentification (SSID) that usually stands for a human-readable string. This guarantees that the positioning unit can identify individual APs even in generic wireless networks such as uni-wireless or eduroam, which are widely adopted by research and education institutions. Finally, it sends the result to the data collector on the cloud server. The server determines the similarity of signal traits, and informs devices in the same cluster with a unique cluster ID.

4.4.1.2 Communication Unit

We adopt and modify Message Queue Telemetry Transport protocol (MQTT) [34]. Its low footprint makes it ideal for mobile and developing Internet of Things (IoT) style applications.
MQTT has publish and subscribe (pub-sub) messaging architecture in which each client indicates a topic to subscribe and sends the subscription to the server. A client publishes messages with a topic, which will be delivered to every client that has the same topic. MQTT makes publishers and subscribers for a single topic communicate on a separate channel. We enable our mobile clients to pub-sub messages using their cluster ID as the topic. By incorporating this functionality, a user can deliver messages to others in the vicinity without explicitly indicating the receivers’ identifier.

4.4.1.3 User Interface Unit

UI aims to support a simple and convenient interface for communication. Unlike other communication applications, our UI has no interface for selecting receivers because the application automatically identifies other users in the vicinity using the cluster ID. Users are allowed to type in plain text, select pictures from the gallery, or copy and paste data from other apps through share with function supported by Android OS as a message body. To protect the message content from publicly viewed by unwanted users, Flock gives a sender the option of entering a password that encrypts the message. To read the encrypted message, the receiver should insert the same password informed offline.

4.4.2 In-cloud Processing

4.4.2.1 Peer Discovery Daemon

Our device detection scheme is implemented as a daemon running on the server. It consists of data collector module, Wi-Fi scan database, similarity computation module, and match table generation module. The data collector receives data from the client and inserts the user ID and Wi-Fi scan results to the Wi-Fi scan database. Then, similarity computation module retrieves scan results from the DB and computes the similarity among them to find devices that have similar signal traits. Based on the similarity measure, match table generation module clusters the devices and allocates a unique cluster ID to each cluster.

4.4.2.2 Communication Unit

Messaging server runs another daemon, an open source MQTT message broker, which communicates with a plurality of mobile devices. It is responsible for managing connections from clients and distributing messages to the interested clients, which are identified by
their cluster ID. Using an object-relational database management system PostgreSQL [43], message DB stores every message received from clients and processes message routing. As we consider messages to be short-lived and volatile, the database retains message data only for a limited amount of time. We define expiration time as 2 hours. This prevents users from receiving messages that are outdated at a specific location, which is identified by the cluster ID.

4.5 Evaluation

In this section, we present evaluation results performed at different sites such as lounges in the library and cafeteria, and the opposite side of the (symmetrical) engineering building to measure the performance in real deployment scenarios.

4.5.1 Discovery Performance

4.5.1.1 Discovery Ratio using Heterogeneous Devices

We characterize the peer discovery performance by two metrics. Discovery ratio represents the number of devices discovered in the same cluster over the number of intended devices for discovery. Success ratio is defined as the percentage of successful experiments out of total experiment attempts. The sensitivity of RSS readings highly depends on the vendor of Wi-Fi...
Table 4.2 Time to get a Wi-Fi scan result and the corresponding peer discovery time for different test devices.

<table>
<thead>
<tr>
<th>Device</th>
<th>Scanning time (s)</th>
<th>Discovery time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung Galaxy S2</td>
<td>4.5</td>
<td>5.9</td>
</tr>
<tr>
<td>Samsung Galaxy Note 2</td>
<td>3.6</td>
<td>4.7</td>
</tr>
<tr>
<td>LG Google Nexus 5</td>
<td>1.6</td>
<td>2.9</td>
</tr>
<tr>
<td>ASUS Google Nexus 7</td>
<td>1.9</td>
<td>3.1</td>
</tr>
<tr>
<td>LG G2x</td>
<td>0.9</td>
<td>2.3</td>
</tr>
</tbody>
</table>

chipset. To reflect the robustness in device heterogeneity, we perform experiments using five different devices from a variety of vendors: Samsung Galaxy S2, Samsung Galaxy Note 2, Google Nexus 5 (LG), Google Nexus 7 (ASUS), and LG G2x.

We select four separate measurement points at an open space in the library and cafeteria. We also choose four measurement points at the corner of each classroom. As there were no cases that all four devices belong to discrete clusters, it is not presented in the result. Figure 4.11 shows that measurements at the library indeed performed the worst; only 80% of trials accurately discovered all target devices. 15% of experiments discovered only half of the devices in proximity. This is not surprising as the experiment was carried out in a relatively large open space, not in a room-size space. On the other hand, experiments conducted in classrooms achieved 100% discovery as expected for all 20 attempts.

4.5.1.2 Discovery Time

The amount of time consumed for discovering others in the vicinity and establishing the connection is another important performance indicator for users in terms of its feasibility as well as practicality. Discovery time includes Wi-Fi scanning time in addition to the networking time for sending the scan results and receiving the cluster ID to and from the server. Time to get a scan result depends on the device, especially on the hardware used. According to our measurement, the time varies from 0.9 seconds (LG G2x) to 4.5 seconds (Samsung Galaxy S2). Table 4.2 summarizes the time to receive a scan result after user’s request and the time to be connected with other peers using various devices. We observe that Wi-Fi scanning time is the main determinant of the discovery operation time. It takes only 2.3 seconds for LG G2x device to discover and connect to its peer. Other devices also finish the operation within
4.6 Conclusion

We prototype Flock system to effectuate a practical but low-cost device interaction interface for smartphones. Our scheme solely analyzes Wi-Fi scan results, which is easily accessible on any commodity mobile devices. To address the challenges of utilizing the dynamic propagation of Wi-Fi signals on heterogeneous devices, we find the time-correlation attribute of RSS measures and leverage the simultaneity of the detection event. We propose a new similarity metric that computes correlation between RSS readings, and present a threshold selection method to be applied on the hierarchical clustering result. Our experimental results demonstrate that the system can accurately cluster and discover co-located devices in general operation scenarios. Accordingly, users can effortlessly discover other devices and communicate with them within seconds.
Conclusion

In this dissertation, we have proposed a practical design of interaction interfaces for mobile devices as a new direction in ubiquitous environments. By adopting an application-driven perspective, this dissertation presents a few useful and novel enhancements. We verify the feasibility of audio-based mobile interaction on commodity mobile devices in spontaneous scenarios. A millimeter-level tracking technique is presented through the sound source localization on two-dimensional surfaces. Additionally, we extend it to three-dimensional motion tracking that can be concurrently used with a vision-based approach in 3D space to compensate for its defects including limited field of view and computation-intensive signal processing. Finally, a proximity detection technique based on the wireless sensing allows context-aware interaction between mobile devices. Although this dissertation leaves many research issues as future work, we believe that it advances the state-of-the-art in the mobile computing field by its contributions in defining, designing, implementing, and evaluating a selection of primitive designs in mobile interactions.
REFERENCES

https://goo.gl/7mX2DR.


http://msis.jsc.nasa.gov/sections/section03.htm.


http://www.myequil.com/home/.


[34] MQTT. http://mqtt.org.


[38] *Number of devices to hit 4.3 per person by 2020*. https://mobileworldlive.com/connected-devices-to-hit-4-3-per-person-by-2020-report.


[47] Samsung Gear VR.


[50] Sony PlayStation Camera.


