APPLYING MULTIMODAL SENSING TO HUMAN LOCATION ESTIMATION

BY

HE WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Nitin Vaidya, Chair
Associate Professor Romit Roy Choudhury, Director of Research
Doctor Dimitrios Lymberopoulos, Senior Researcher, Microsoft Research
Professor Klara Nahrstedt

ABSTRACT

Mobile devices like smartphones and smartwatches are beginning to “stick” to the human body. Given that these devices are equipped with a variety of sensors, they are becoming a natural platform to understand various aspects of human behavior. This dissertation will focus on just one dimension of human behavior, namely “location”. We will begin by discussing our research on localizing humans in indoor environments, a problem that requires precise tracking of human footsteps. We investigated the benefits of leveraging smartphone sensors (accelerometers, gyroscopes, magnetometers, etc.) into the indoor localization framework, which breaks away from pure radio frequency based localization (e.g., cellular, WiFi). Our research leveraged inherent properties of indoor environments to perform localization. We also designed additional solutions, where computer vision was integrated with sensor fusion to offer highly precise localization. We will close this thesis with micro-scale tracking of the human wrist and demonstrate how motion data processing is indeed a “double-edged sword”, offering unprecedented utility on one hand while breaching privacy on the other.

To Mom, Dad, and Yanchi

ACKNOWLEDGMENTS

I am deeply indebted to many:

To my advisor, Romit Roy Choudhury, for his guidance in my Ph.D. journey, and, more importantly, for changing my life. He has always been there to help me throughout my Ph.D. – at my Ph.D. milestones, during our research demonstration to Intel, and in my career decisions. I will never forget the late nights before paper submissions and he always said, "He, go to sleep. You need a fresh mind tomorrow. I will give you this part before you wake up." He inspired me in the research philosophy of exploring both bottom-up and top-down approaches and encouraged me pursuing my academic career. Without his encouragement and support, I would not have gotten nearly this far.

To the University of Illinois at Urbana-Champaign Electrical and Computer Engineering and Computer Science departments for all the support during my Ph.D. study. I especially thank Nitin Vaidya and Klara Nahrstedt for serving on my doctoral committee, and for their insightful comments.

To my committee member and internship mentor Dimitrios Lymberopoulos, for his constant support in many facets of my Ph.D. career. To many others at Microsoft Research, especially Jie Liu and Jacky Shen. To Srihari Nelakuditi, for his encouragement and help.

To all my friends and labmates. To Souvik Sen and Xuan Bao, for their generous help in my early years of research.

To my other co-authors, Moustafa Youssef, Ted Tsung-Te Lai, Nirupam Roy, and Sheng Shen.

To ECE Publications Office, Jan Progen in particular, for going through this dissertation and for helping me improve its quality and readability.

To my wife, Yanchi and my parents, for their unconditional love, unwavering support, and faith in me.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In a matter of 10 years, from 2005 to 2015, the mobile phone has transformed from a basic communication device to a smart device that performs sensing, computing, and communications. Given that these smart devices are always on and always with us, they are envisioned as a general-purpose human sensor, capable of zooming into our daily lives and understanding our activities, preferences, and behavioral patterns. The Silicon Valley has been calling this the "quantified self" [1], essentially suggesting that the data from these devices can be used for core inferences about ourselves, ultimately enabling a wide variety of mobile sensing applications.

Many mobile sensing applications are already in the market. For example, using accelerometer data from smartphones, it is possible to count the number of steps the user has walked. Various calorie score applications have emerged [2, 3, 4, 5]. Newer smartwatches have skin conduction sensors that read the heart rate and enable various mobile health applications [2, 3, 4]. The Global Positioning System (GPS) provides driving direction.

While these were initially innovative and important, now they are common. The users of the next generation of applications have much higher expectations. If we need to deliver better applications for the future, research is necessary to deliver efficient, practical inferencing techniques for humans. For instance, finding a person's indoor location, estimating the posture of a person, tracking hand gestures, and various forms of context-awareness are four inferences that are still difficult to assess today.

Our research focuses on developing inferencing techniques using multimodal sensor data on mobile devices such as smartphones and smartwatches, with an emphasis on location sensing and tracking.

In recent years, numerous location-based services have emerged in smartphone application stores. These location-based services significantly improve people's life quality. However, the vast majority of these services are only available for outdoor scenarios. This is mainly attributed to the comparative immaturity of indoor localization technology. While GPS has revolutionized outdoor localization, it does not work indoors. Thus, indoor localization has been a tantalizing problem in mobile computing, and despite significant research, there is no solution yet in the mainstream. Past work has explored signals from external infrastructures (such as WiFi, sound, Bluetooth), however, installation, maintenance, and periodic calibration needed in such infrastructure-based solutions render them impractical.

Around 2007, motion sensors such as the accelerometer, gyroscope, and compass, were integrated onto smartphones, which opened up the possibility of using them for localization as opposed to relying on external infrastructures. A few schemes have demonstrated the ability to compute the motion trajectory of a mobile phone, using the phone's speeds estimated from its accelerometer and compass. The general idea is calculating one's current location by using a previous location, and advancing that position based upon estimated speeds over time. This has been called dead-reckoning. Since the estimated speeds are not alway correct, the dead-reckoned trajectories are accurate in the beginning, but diverge from the correct trajectories over time. However, the accuracy of dead-reckoning can be increased significantly by using reference points to obtain fixes occasionally. For instance, in the past, stars were used as reference points to guide pilots in dead-reckoning based air navigation systems.

Our research, UnLoc [6], an unsupervised indoor localization scheme, identifies indoor reference points, uses these reference points to reset dead-reckoning errors and provides highly accurate indoor localization. Our key observation is that certain locations in an indoor environment present identifiable signatures on one or more sensing dimensions. An elevator, for instance, imposes a distinct pattern on a smartphone's accelerometer; a specific spot may experience an unusual magnetic fluctuation; a corridor-corner may overhear a unique set of WiFi access points. We hypothesize that these kinds of signatures naturally exist in the environment, and can be envisioned as internal landmarks of a building. Mobile devices that "sense" these landmarks can recalibrate their locations, while dead-reckoning schemes can track them be-

tween landmarks. Translating this idea-sketch into a functional system entails a variety of challenges such as *(i) how to identify landmarks without human supervision; (ii) how to automatically estimate landmark locations; (iii) how to bootstrap the system without manual calibration effort; (iv) how to overcome the impact of indoor magnetic fluctuation on walking direction estimation.* UnLoc systematically addressed these challenges and achieved a median localization accuracy of 1.69 m. A video demonstration of UnLoc can be found in [7]. UnLoc is discussed in Chapter 2.

UnLoc achieves a high median accuracy of 1.69 m without infrastructure and calibration costs; however, the tail of the error distribution is long. In some applications, 99th percentile accuracy should be high and people are willing to pay the expense for such accuracy. Therefore, we also studied solutions using some infrastructures, VideoLoc, where feeds from surveillance cameras can be leveraged for highly precise localization, without compromising the privacy of individual users. While computer vision technologies can estimate a user's location from surveillance cameras, the key question in VideoLoc is how to send the estimated location from the camera to the user's smartphone. To enable such communication, we need the user's visual address. While the user's face could be one possible approach, the face may not be always visible. Our core technique [8, 9] exploits the intuition that human motion patterns and clothing colors can together encode several bits of information. By treating this information as a "visual address", it is feasible to enable communication between a camera and a user, while allowing the user to turn off the "visual address" at will. This core technique in VideoLoc can be extended to other applications such as augmented reality. VideoLoc is discussed in Chapter 3.

Using rich sensors on mobile devices to track human location enables important applications. Such data is often a "double-edged sword" since it leaks information about aspects of lives that are considered private. We looked into the potential information leaks from micro-scale tracking of the human wrist using wearable devices. Imagine a user typing on a laptop keyboard while wearing a smartwatch. We asked whether motion sensors from the watch can leak information about what the user is typing [10]. While it is not surprising that some information will be leaked, the question is how much? We found that when motion signal processing is combined with patterns in English language, the leakage is substantial. Reported results showed that when a user types a

word W that is longer than six characters, it is possible to shortlist 10 words on average that will include W. The primary technical challenges are (i) tracking micro motion of the smartwatch, and (ii) inferring the timing and location of key-presses using (noisy) motion sensors. We addressed these challenges with sensor fusion, signal processing and machine learning techniques. We named our project MoLe, which stands for "motion leaks". A brief video introduction to MoLe can be found in [11]. MoLe is discussed in Chapter 4.

CHAPTER 2

UNLOC: UNSUPERVISED INDOOR LOCALIZATION

We propose *UnLoc*, an unsupervised indoor localization scheme that bypasses the need for war-driving. Our key observation is that certain locations in an indoor environment present identifiable signatures on one or more sensing dimensions. An elevator, for instance, imposes a distinct pattern on a smartphone's accelerometer; a corridor-corner may overhear a unique set of WiFi access points; a specific spot may experience an unusual magnetic fluctuation. We hypothesize that these kinds of signatures naturally exist in the environment, and can be envisioned as internal *landmarks* of a building. Mobile devices that "sense" these landmarks can recalibrate their locations, while dead-reckoning schemes can track them between landmarks. Results from three different indoor settings, including a shopping mall, demonstrate median location errors of 1.69 m. War-driving is not necessary, neither are floorplans – the system simultaneously computes the locations of users and landmarks, in a manner that they converge reasonably quickly. We believe this is an unconventional approach to indoor localization, holding promise for real-world deployment. This chapter is published in MobiSys 2012 [6].

2.1 Introduction

Despite innovative research [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26], indoor localization is still not in the mainstream. In trying to trace the reasons, we distilled two main messages: (1) Indoor spaces require fairly high location accuracy because the contexts vary at finer spatial granularity. For instance, a 5 m location error outdoors may still indicate the same street, but 5 m indoors may mean two different aisles in a grocery store. An inventory-management application may very well require aisle-level precision. (2) While such precision is attainable with pervasive WiFi systems, they come at

a prohibitively high cost, mostly in the form of meticulous (signal) calibration. Such calibration is not necessarily a one-time cost since RF fingerprints could change, perhaps due to changes in layout and objects in the environment. Attempts to simplify the calibration process have been successful, but at the expense of reduced location accuracy. This zero sum game between accuracy and calibration overhead has been an important hurdle to deploying indoor localization systems. This chapter is tasked to break away from this tradeoff, and achieve *meter* level accuracy with zero calibration. Although a high bar, we believe this is feasible and pursue this goal.

Our scheme cuts across isolated ideas in mobile computing. We introduce these ideas first, and then describe the value in making them compatible.

- A few recent schemes have demonstrated the ability to compute the motion trajectory of a mobile phone, using its accelerometer and compasses [13, 27]. This has been called urban dead-reckoning. Due to noise in the mobile sensors, the dead-reckoned trajectories are accurate in the beginning, but diverge from the correct trajectories over time. Therefore, outdoor localization schemes like CompAcc [13] have triggered periodic GPS measurements to recalibrate the user's location. Unfortunately, GPS is unreliable indoors, rendering dead-reckoning based approaches useless. Nonetheless, if one can identify other means of recalibration, dead-reckoning could be applicable even indoors.

- Urban sensing and activity recognition literature have demonstrated the ability to recognize ambiences and user behavior. For instance, inertial sensors can detect when a user is walking, turning into a corridor, or climbing up the stairs [28]; microphones and magnetometers can detect ambient sounds and magnetic fluctuations [25, 29]. While these signatures have been primarily used for various forms of context-awareness, they lend themselves to localization as well. For example, these signatures can be treated as "landmarks", useful to recalibrate indoor dead-reckoning.

- Past work has mostly relied on signal calibration [12, 14] to develop WiFi-based localization. We observe that WiFi can be valuable even without calibration. For instance, one can use overheard WiFi APs to partition an indoor space into smaller sub-spaces. Thus, a landmark signature need

not be unique in the entire building – so long as it is unique within a WiFi sub-space, it can be recognized without ambiguity.

UnLoc combines these three ideas (dead-reckoning, urban sensing, and WiFi-based partitioning) into a framework for unsupervised localization. The core idea is recursive but not complicated. Briefly, mobile users move naturally in the building collecting accelerometer, compass, gyroscope, and WiFi readings. By assimilating data from these devices, UnLoc detects sensory signatures (e.g., a corridor turn) that are unique within their respective WiFi sub-spaces. Now using the same collected data, UnLoc dead-reckons the devices starting from a known reference location, say the entrance of the building. Since dead-reckoning provides a rough location to the phone, it is also possible to roughly localize the signatures based on when the phone senses them.

Now, the locations of these signatures – also called *landmarks* – can be made more accurate by combining the rough estimates from multiple phones. These landmarks can then be used to improve the dead-reckoning of subsequent phones, which in turn can refine the landmark locations. This recursive process continues to improve localization accuracy over time. Observe that the system does not need calibration – the first few users may experience inferior location accuracy, but a little more data brings the system to convergence.

Of course, translating this idea-sketch into a functional system entails a variety of challenges: (1) Dead-reckoning is non-trivial in indoor environments where metals and electrical equipment significantly affect the compass bearing.[1] (2) Extracting the accurate location of the landmarks from multiple erroneous locations is problematic, since all the errors may not be equal. Some notion of confidence on the errors needs to be built, so that the estimates can be suitably weighted before combination. (3) Identifying landmark signatures from the sensed data warrants unsupervised learning on sensor features. (4) Finally, the system needs to be optimized for energy, to avoid a significant battery drain during localization. This chapter addresses these challenges one step at a time, prototypes on a testbed of Android Nexus series phones, and evaluates with three volunteers naturally walking in two university buildings and one shopping mall. Performance results show median localization accuracy of 1.69 m when UnLoc runs online, and 0.89 m when the locations are computed of-

[1] Prior work demonstrated dead-reckoning mostly for outdoor environments.

fline. The system quickly reaches convergence in less than two man-hours, and remains robust to dynamic changes in the environment. We believe this could be a promising direction, and with rigorous testing and tuning, a potential candidate for the real world.

Our main contributions may be summarized as follows:

- **We identify an opportunity to simultaneously harness sensor-based dead-reckoning and environment sensing for localization.** Our approach does not require calibration or installation of additional infrastructure.
- **We design a practical scheme that employs unsupervised learning to extract unique sensor signatures – called landmarks.** We show that adequate landmarks exist inside buildings such that dead-reckoning is practical and reasonably accurate.
- **We develop UnLoc on the Android OS, and evaluate across three different indoor spaces, including the North Gate shopping mall in Durham.** We achieved less than 2 m error without any pre-deployment effort; in fact, we used the map of the building only to compute the ground truth for evaluation.

The subsequent sections expand on each of these contributions, beginning with an architectural overview and intuition, followed by measurement, design, and evaluation.

2.2 Architecture and Intuition

We begin with a high-level overview of UnLoc, focusing mainly on the core building blocks and intuitions (Figure 2.1). We will postpone the discussion on actual algorithms and engineering details to Section 2.3. In fact, *we will even make a simplifying assumption that the building floorplan is available to UnLoc.*

Of course, this assumption need not hold in our final system – its sole purpose is ease of explanation. Once we have developed the core framework, we will discuss how the assumption can be relaxed.

Human Motion Traces Consider the example where we intend to localize users in a shopping mall. When users visit the mall, the UnLoc app running on their smartphones collects time-stamped sensor readings. These readings (mainly from accelerometer, compass, gyroscope, and WiFi APs[2]) are assimilated in a central repository. UnLoc must operate on this data to track each user's location. For now, we perform this offline.

Seed Landmarks (SLMs) As a first step, UnLoc looks into the floorplan of the building and identifies some "*seed landmarks*". These seed landmarks are essentially certain structures in the building – stairs, elevators, entrances, escalators – that force users to behave in predictable ways. These predictable behaviors can be translated to sensor signatures. For instance, building entrances are characterized by a visible drop in the GPS confidence when the user moves from outdoors to indoors; elevators exhibit a distinct accelerometer signature, emerging from the start and stop of the elevator. If a user, Alice, used the elevator in the mall, UnLoc expects her trace to contain the elevator signature embedded in it. Let us assume that this signature occurs in Alice's trace at time t_i. Since the location of the elevator is known within the floorplan, UnLoc can precisely localize Alice at t_i. In similar ways, Alice can be precisely localized at other time-points when she passed through any of the other landmarks. The next step, then, is to localize her while she moved between these landmarks – for this, UnLoc adopts techniques from urban dead-reckoning.

Dead-Reckoning Urban dead-reckoning [13] is an established idea that uses the accelerometer and the compass to track a mobile user. The key idea is simple. Based on the accelerometer readings of the mobile phone, it is possible to count the number of steps a person has walked, and from there derive the displacement of the person. Based on the compass, the direction of each of these steps can be tracked.[3] Merging these, the <*displacement, direction, time*> tuple forms the human's motion vector. Pivoting these vectors at the seed landmarks, UnLoc tracks the location of Alice. Although the tracking operation is crude (because the noisy sensors accumulate error over time), the error gets reset whenever Alice crosses any of the landmarks. Thus, in the steady state,

[2]Sound and light measurements could also be useful, but we do not activate them for energy-efficiency.

[3]However, as shown in Section 2.2.1, magnetic fluctuations in the environment will derail the compass readings, forcing us to shift to gyroscopes.

Alice's localization error exhibits a saw-tooth behavior – over time, the localization error grows and then sharply drops to zero at the landmark, and then grows again. Observe that increasing the density of landmarks will cause the error to reset frequently, thereby curbing the error growth. UnLoc will attempt to accomplish this by organically extracting additional landmarks from the indoor environment.

Organic Landmarks (OLMs) In addition to seed landmarks, UnLoc postulates that any indoor environment will offer some ambient signatures across one or many sensing dimensions. These signatures can be in the magnetic domain, wherein metals in a specific location may produce unique and reproducible fluctuations on the user's magnetometer, near that location. Signatures could also be WiFi-based – a spot may overhear a set of WiFi base stations, but the set may change at short distances away from that spot. A few (dead) spots inside a building may not overhear any WiFi or GSM/3G signals, which by itself is a signature. Further, even a water-fountain could be a signature – users who stop to drink water may exhibit some common patterns on the accelerometer and magnetometer domains. Whenever these pattern surface on Alice's trace, it could be an opportunity to recognize her location. Of course, even though these signatures can become useful landmarks, they cannot be known *a priori*, and will vary across different buildings. They have to be learned dynamically.

Toward learning these landmarks, UnLoc subjects the sensor data (gathered from all phones) to a clustering algorithm (Figure 2.1). Actually, various features of this data are extracted and the clustering runs on this high-dimensional space – detailed in Section 2.3. Once the clustering operation has completed, each of the resulting clusters is expected to contain similar sensor patterns. Now, since each sensor reading is associated with time-stamps, it is possible to find their corresponding locations via dead-reckoning. UnLoc computes these locations to check whether all members of a cluster fall within a small area as shown in Figure 2.2. If they do, then UnLoc deems it a new landmark. Since these landmarks were discovered automatically, without any external supervision, we call them *organic landmarks* (OLMs). If no OLMs are found, UnLoc waits for more traces and continues scanning for OLMs.

Simultaneous Localization and Mapping Both SLMs and newly discovered OLMs are used to improve dead-reckoning for subsequent users, which in turn improves the location estimates of the SLMs/OLMs themselves. This circular process pushes the entire system to better accuracy and UnLoc continues to improve over time. Section 2.4 will quantify this behavior for different users and buildings.

2.2.1 Intuition and Supporting Measurements

The success of UnLoc hinges on at least three performance-related expectations: (1) Dead-reckoning can attain desired levels of accuracy, if periodically recalibrated by landmarks. (2) Indoor environments indeed offer the requisite number of landmarks. (3) The locations of the landmarks can be computed from rough estimates of multiple devices (i.e., the dead-reckoning errors are indeed independent). We discuss our intuition in each of these, and present supporting measurements.

2.2.1.1 Dead-Reckoning Accuracy

We performed some initial experiments in the computer engineering building at Duke University. Volunteers were asked to carry NexusS phones in their pockets, and walk naturally in one wing of the building. The UnLoc app running on the phones records the accelerometer and compass readings, and extracts from them the <*displacement, direction, time*> tuples. To record ground truth, we marked different doors and windows in the buildings with a distinct number[4] – when a user passed by that number, the user entered it into the user's phone. Since we know the mapping between the number and the door/window, we are able to extract the ground truth (almost accurately). We gathered 10 traces, each starting from the entrance of the building. We performed trace-based analysis to understand how the dead-reckoned path diverges from the true path, with varying number of landmarks.

Figure 2.3 shows the accumulated error over time (light gray curve) when using pure dead-reckoning with zero landmarks. Evidently, the error accumulates dramatically fast, and is completely unusable. Hence, we simulate

[4] Recall that GPS is unavailable indoors.

landmarks by periodically reseting the user's location to the correct location – the black curve shows the results. While the performance improves, the mean localization error is still 11.7 m, greater than our target. On analyzing the data, we observed that the magnetic field in indoor environments is heavily distorted by metallic and electrical equipments in the (engineering) building. Thus, although these magnetic fluctuations are beneficial for finding landmarks, they derail dead-reckoning by injecting heavy error in the compass. UnLoc appeared impractical at this point.

Fortunately, newer smartphones are embedded with gyroscopes that measure the angular velocity of the phone in three dimensions and are not affected by the magnetic field. We carefully processed the gyroscope readings to compute the angular changes during walking, i.e., how the user turned. However, gyroscope readings are relative and we needed to combine it opportunistically with the compass to estimate the user's absolute walking direction. We will report several technical challenges of this operation in Section 2.3, including compass offsets and gyroscope drifts. However, once we harnessed the gyroscope and the compass in tandem, the dead-reckoning error reduced appreciably (blue curve in Figure 2.3). When re-calibrated by periodic landmarks, the average error dropped further to 1.2 m, offering us confidence to build a fuller system.

In addition, once a landmark is encountered, the user's path can be retraced and corrected between the last two landmarks. Although this does not help in real-time tracking of the user, it helps in offline analysis, and more importantly, for improving the location estimate of organic landmarks.

2.2.1.2 Landmark Density

It is natural to question *why indoor environments would exhibit sensor signatures to be used as landmarks.* While there is no guarantee, our observation is that an indoor environment is rich with ambient signals, like sound, light, magnetic field, temperature, WiFi, 3G, etc. Moreover, different building structures (e.g., stairs, doors, elevators) force humans to behave in specific ways. If one "combs" through all these sensor signals and their high-dimensional combinations, we postulate that some signatures are likely to emerge. The intuition is essentially rooted in diversity, i.e., the chances that all of the signals are similar

and no pattern "shows up as different", seems unlikely. Further, these signatures need not be unique in the entire building – so long as they are unique within the WiFi sub-space, they can be valid landmarks. We performed a small scale measurement study to verify this intuition. The results follow.

WiFi Landmarks Consider an area within which all locations overhear a distinct set of WiFi APs. An indoor space is likely to have many such areas of varying sizes. Figure 2.4 shows the distribution of the sizes of these areas. While most of the areas are quite large – explaining why simple WiFi-based localization is not very accurate – there are a few areas (at the left side of the X-axis) that are very small. UnLoc aims to exploit these small areas as a landmark. If one of these small areas overhears a set of WiFi APs, denoted W, then a mobile phone overhearing the same set can be assumed to be within that area. Since the area is small, the localization error of the phone will be small too, enabling a location recalibration. Our measurements show that in two floors of the engineering building, we find eight and five such WiFi landmarks, each of area less than 4 m^2. Thus WiFi APs can offer landmarks to enhance dead-reckoning.

Magnetic/Accelerometer Landmarks In search of signatures in other sensing domains, we executed K-means clustering on accelerometer and compass measurements (we present the algorithm and parameter details in Section 2.3). For each cluster, we mapped their members to their corresponding physical locations (using ground truth). For most clusters, we found that their member-locations were widely scattered in space, and hence, were unusable as a landmark. However, members of a few clusters proved to be tightly collocated in space as well. For example, we discovered a unique/stable magnetic fluctuation near our networking lab. We found another spot with a distinct accelerometer signature – a pair of symmetric bumps in opposite directions (Figure 2.5).

Although UnLoc need not understand the semantic meaning of these signatures, out of curiosity we analyzed the data, and discovered that they were caused by the elevator starting and stopping. In fact, the direction of the bumps (upward or downward) even indicated whether the user went upstairs or downstairs. These spatially collocated patterns were natural landmarks, and when we assimilated all sensing dimensions, the number of landmarks proved to be

six and eight for each floor. In fact, when we combined accelerometer and compass together to create a higher-dimensional signature, we found even more landmarks due to turns in the building. In sum, the organic and seed landmarks together seemed to offer the needed density to support indoor dead-reckoning.

2.2.1.3 Computing Landmark Locations

Perhaps an important pitfall in UnLoc lies in computing the locations of the landmarks. Figure 2.6(a) is intended to explain the problem. In this example, assume that UnLoc has combined the user's sensor data, clustered on the data points, and discovered three sensor signatures (in distinct WiFi areas), that can be used as landmarks. Now the locations of these landmarks need to be computed, but there is no ground truth to learn that information. One way to estimate the location of a landmark is to use dead-reckoning, but the result will not be accurate since dead-reckoning itself is erroneous. Our approach is to compute the landmark location by combining all the (dead-reckoned) estimates of a given landmark. The intuition is that dead-reckoning errors have been observed to be random and independent, due to the noise in hardware sensors and human step sizes [13]. By combining these errors from adequate measurements, one could expect the estimated mean to converge to the actual landmark location. Figure 2.6(b) illustrates the opportunity through a simple centroid calculation.

Figure 2.7 visualizes data from real measurements to verify independence in dead-reckoning error in which each line joins the estimated landmark location to the actual location. Visually, the errors appear uncorrelated. Of course, this is a preliminary overview of the inner workings of the system. We later discuss the algorithmic details of this error combining process, and measure performance in greater detail.

2.3 Design Details

This section describes the algorithms and engineering details underlying UnLoc.

2.3.1 Seed Landmarks

If the building's floorplan is known (which is often necessary to visualize the user's location), then UnLoc can infer the locations of doors, elevators, staircases, escalators, etc. This implies that the location of seed landmarks (SLMs) are immediately known. As long as the smartphone can detect these SLMs while passing through them, it can recalibrate its location. Thus, the goal of the SLM detection module is to define sensor patterns that are global across all buildings.

2.3.1.1 Elevators, Staircases, and Escalators

This class of SLMs is based on using the inertial sensors. These sensors have the advantage of being ubiquitously installed on a large class of smartphones, have a low-energy footprint, and are always turned on during the phone operation (to detect the change of screen orientation). We focus on three particular examples that are common in indoor environments: elevators, escalators, and stairs. Figure 2.8 shows a classification tree for detecting the three classes of interest and separating them from walking and being stationary. We note that a false positive leads to errors in estimating the location of the SLM while a false negative leads to missing an opportunity for recalibration. Therefore, high detection accuracy with low false positive/negative rates are highly desired.

Elevator A typical elevator usage trace consists of a normal walking period, followed by waiting for the elevator for some time, walking into the elevator, standing inside for a short time, an over-weight/weightloss occurs (depending on the direction of the elevator), then a stationary period which depends on the number of the floors the elevator moved, another weight-loss/over-weight, and finally a walk-out. To recognize the elevator motion pattern, we developed a Finite State Machine (FSM) that depends on the observed state transitions. Different thresholds are used to move between the states. Evaluation over 22 traces shows that the thresholds are robust to changes in the environment and can achieve 0.6% and 0% false positive and negative rates, respectively.

Escalator Once the elevator has been separated, it is easy to separate the classes with constant velocity (escalator and stationary) from the other classes

(walking and stairs) using the variance of acceleration. Now, to separate the escalator from stationarity, we found that the variance of the magnetic field can be a reliable discriminator. Of course, a user may sometimes climb up the escalator – we find that magnetic variance also differentiates between this and an actual staircase-climb (see Figure 2.9).

Stairs Once the scenario with constant speed is separated, we need to differentiate between the stair and walking cases. The main observation here is that when the user is using the stairs, the user's speed increases or decreases based on whether the gravity is helping the user or not. This creates a higher correlation between the acceleration in the direction of motion and direction of gravity as compared to walking. As reported in Section 2.4, staircases can sometimes lead to false negatives (1.8%).

2.3.2 Dead-Reckoning

The two sub-tasks in dead-reckoning are (1) computing the user's displacement from the accelerometer and (2) continuously tracking the direction of movement.

2.3.2.1 Displacement from Accelerometer

One possible solution is to double-integrate the accelerometer readings. Figure 2.10(b) shows the unacceptable results, that is, the difference between the estimated and actual displacement reaches more than 100 m only after 30 m of actual displacement. This is an attribute of a noisy accelerometer, low sampling rate (24 Hz), as well as jerky movements of the phone when carried by the user. A better approach [13, 30] is to identify a human-walking signature as in Figure 2.10(a). This signature arises from the natural up/down bounce of the human body for each step taken. To capture this, we pass the signal through a low pass filter, and identify two consecutive local minima. Between these local minima, we search for a local maxima, and check whether the difference between the maxima and minima is greater than a threshold. If so, we increment the step count.

The physical displacement can be computed by multiplying the step count with the user's *step size*, a function of the user's weight and height [31].[5] Employing a fixed step size across all users can clearly be erroneous. However, UnLoc has the opportunity to infer the step size by counting the number of steps for a known displacement (i.e., between two landmarks). Figure 2.10(c) shows the error accumulated by UnLoc using these techniques in place – the results are encouraging. The step count accuracy (verified with 10 users) was also 98%.

2.3.2.2 Orientation Using Compass/Gyroscope

Past work has demonstrated the feasibility of dead-reckoning using the smartphone compass. However, to the best of our knowledge, all these results are for outdoor environments. Indoors, the magnetic field due to ferromagnetic material and electrical objects in the vicinity, completely derailed our dead-reckoning attempts.[6] In response to this, we explored the feasibility of using a gyroscope to *infer* the user's movement directions. Our intuition was that the gyroscope is decoupled from the magnetometer sensor, and hence, could be insensitive to ambient magnetic fields.

While this insensitivity proved true, i.e., the gyroscope indeed remained unaffected by changing magnetic fields, the tradeoff was that the gyroscope offered *relative angular velocity*. This is in the form of a 3D rotation matrix which when multiplied by a time interval, yields the relative angular displacement (RAD) of the device. Unfortunately, the RAD is with respect to a direction that is not necessarily the absolute direction. Thus, while we could track the structure of the user's motion path using the gyroscope, these paths were biased by the error in their initial direction. Thus all the estimated paths appeared as rotated versions of the true path, shown in Figure 2.11(a).

We observe that encountering landmarks can help infer this bias. Figure 2.11(b) explains the opportunity. Consider a user encountering a known landmark L_1 at time t_1, and later another landmark L_2 at time t_2. UnLoc identifies that the user encountered these two landmarks because the signatures matched, and hence, regardless of the dead-reckoned estimates, UnLoc "pins down" the user's path at these locations. Now, let θ denote the angle between

[5] A more accurate approach is to estimate the user step size based on the user's gait [32].

[6] We attempted at least five different techniques to learn and correct for these fluctuations.

the line joining L_1, L_2 and the line joining L_1, X_2, where X_2 is the dead-reckoned estimate at time t_2. We observe that θ is the initial bias, and therefore, UnLoc rotates the entire motion segment by θ. Importantly, the same θ can be used to track the user for the subsequent motion segment – say until the user encounters landmark L_3 – at which point the bias can again be updated. This process of learning and updating the bias at every landmark leads to stable and consistent results. The only remaining problem lies in tracking the user until the second landmark is encountered, and during this phase, the user's dead-reckoning error can be arbitrarily high.

UnLoc turns to the compass during the initial phase when the gyroscope bias is still unknown. Clearly, the compass value cannot be used blindly – ambient magnetic field fluctuations can actually degrade the results. Thus, UnLoc juxtaposes the gyroscope and compass readings, shown in Figure 2.12(a), and whenever the trends are correlated, the compass value is selected as the direction of motion. The gyroscope's bias is now inferred, and used thereafter. The intuition here is that correlated trends in the compass and gyroscope are an indicator of proper compass readings; and if the compass is not reflecting the gyroscope's trend, it is probably affected by other factors. Figure 2.12(b) shows the eventual outcomes in an angular direction estimation. The compass helps with the initial phase, while the gyroscope-based dead-reckoning proves to be effective. In fact, to the best of our knowledge, UnLoc is the first system to leverage smartphone gyroscopes for indoor dead-reckoning.

2.3.3 Organic Landmarks

The task of discovering organic landmarks (OLMs) is rooted in (1) recognizing distinct patterns from many sensed signals, (2) and testing whether a given pattern is spatially confined to a small area. Recall that Figure 2.2 illustrates the flow of operations. All the sensor readings are gathered in a matrix: element $< i, j >$ of the matrix contains sensor readings from phone i at time j. These sensor readings are essentially *features* of the raw sensed values (from the accelerometer, compass, gyroscope, magnetometer, and WiFi). Features for the magnetic and inertial sensors include *mean, max, min, variance, mean-crossings*, while for WiFi, they are *MAC ID, RSSI*.

UnLoc normalizes these features between $[-1,1]$ and feeds them to a $K-means$ clustering algorithm. (We tested with *Expectation Maximization* (EM) clustering as well, but it was not consistently better than K-means.) The clustering is executed for each individual sensing dimensions, as well as their combinations (such as accelerometer and compass together). Figure 2.13, for example, shows the clusters from the magnetometer readings for $K = 3$. We varied the value of K and recorded the clusters in each case. Our goal is to identify clusters that have *low similarity* with all other clusters; this will suggest a good signature. For this, we compute the correlation between a given cluster and all other clusters. If the maximum correlation is less than a *similarity threshold*, we consider this cluster as a candidate for a landmark.

To qualify as an OLM, the candidate cluster must also be confined to a small geographical area. For this, we first test whether the members of a cluster are within the same WiFi area (i.e., they overhear the same WiFi APs). While this is necessary, it is not sufficient because many WiFi areas are large. Therefore, for clusters within a WiFi area, we compute the dead-reckoned locations for each of their members. If locations of all cluster-members are indeed within a small area – we use $4\,m^2$ – then we recruit this cluster as an OLM. As an anecdote, we found that using the accelerometer, one of the sensor clusters was scattered all over the indoor space. Upon investigation, we detected that this cluster roughly captured walking patterns. On the other hand, another cluster that proved to be within the $4\,m^2$ area was from a magnetic signature near an electrical service room in the building. UnLoc announces the centroid of the cluster as the OLM's location.

While the above describes the generalized version of the OLM detection algorithm, the different sensing dimensions require some customization, discussed next.

2.3.3.1 WiFi Landmarks

We use MAC addresses of WiFi APs and their corresponding RSSI values as features. To remain robust to signal variations (which alters the set of overheard APs), we only consider APs that are stronger than a threshold RSSI. Now, applying K-means clustering, we identify small areas ($4\,m^2$) that have low similarity with all locations outside that area. We compute the similarity of two locations,

l_1 and l_2, as follows. Let us denote the sets of WiFi APs overheard at locations l_1 and l_2 as A_1 and A_2, respectively. Also, let $A = A_1 U A_2$. Let $f_i(a)$ denote the RSSI of AP a, $a \in A$, overheard at location l_i; if a is not overheard at l_i, then $f_i(a) = 0$. We now define similarity $S \in [0,1]$, between locations l_1 and l_2 as:

$$S = \frac{1}{|A|} \sum_{\forall a \in A} \frac{min(f_1(a), f_2(a))}{max(f_1(a), f_2(a))}$$

The rationale for this equation is to add proportionally large weights to S when an AP's signals are similarly strong at both locations, and vice versa. We threshold on S to define a WiFi landmark. We choose a threshold of 0.4 in our system, indicating that all locations within the WiFi landmark need to exhibit less than 0.4 similarity with any other location outside the landmark. Of course, this rather strict threshold ensures that landmarks are quite distinct, but also reduces the number of possible landmarks. Figure 2.14 shows this tradeoff using traces from two Duke University buildings. We observed that 0.4 was a reasonable cut-off point, balancing quality and quantity of WiFi OLMs.

2.3.3.2 Magnetic and Inertial Sensor Landmarks

Indoor environments are characterized by at least a few turns (at the end of corridors, into offices, classrooms, stairs, etc.). Since the gyroscope offers reliable angular displacements, we recognize the opportunity to use them as organic landmarks. We design a special feature called the *bending coefficient*. Essentially, the coefficient captures the notion of path curvature, computed as the length of the perpendicular from the center of a walking segment to the straight line joining the end-points of the segment. We compute the bending coefficient over a sliding window on the user's walking path, and use them as a separate feature. Later, when we cluster on the bending coefficient and WiFi together as features, similar turns within a WiFi area gather in the same cluster. The turns in the cluster could still be doors of adjacent classrooms in a corridor – these turns may very well lie within the same WiFi area. To avoid coalescing all these turns into the same landmark, UnLoc checks if the cluster is confined to within a 4 m^2 area; only then is the cluster declared a landmark.

Magnetic landmarks are also derived through similar techniques. So long as the magnetic signature is unique within one WiFi area, and the sensed locations

are spatially confined within $4\,m^2$, we deem it as a magnetic OLM. Figure 2.15 shows an anecdotal example where the magnetic field near our networking lab demonstrates a unique distortion.

2.3.4 Simultaneous Localization and Mapping

UnLoc uses the SLMs and OLMs to reset dead-reckoning error and track the user. The improved dead-reckoned paths help in refining the landmark locations because different paths offer *independently* erroneous estimates of a specific landmark. Combining these independent errors produces the refinement (the intuition derived from the law of large numbers, where the sampled mean converges to the true mean for a large number of samples). We combine the estimates of a landmark, say L_i, as follows. While the obvious approach would be to compute the centroid, we actually take advantage of the observation that all estimated locations may not be equally incorrect. Consider two users who arrive at L_i from landmarks L_j and L_k, respectively. If L_j is closer to L_i than L_k, then the user who walks from L_j is likely to have incurred less error. This is because pure dead-reckoning is known to accumulate error over time. Thus, accounting for this confidence in landmark estimates, UnLoc computes a weighted centroid. The result is declared as the location of the landmark.

2.3.5 Points of Discussion

While we assumed the knowledge of a floorplan, we can relax that assumption now. Observe that in reality, we need just one ground truth location of any seed landmark. This could be the location of the building's entrance, staircase, elevator. Once we know the GPS coordinate of one SLM, the rest of the SLMs and OLMs can be organically grown, using this known coordinate as the origin. Importantly, the location of this origin SLM – say the building's entrance – needs to be learned *only once* when UnLoc bootstraps for the first time. In the steady state, even if users do not know when they are passing through the entrance, their locations will become known once they encounter a landmark. The same is true when a user enters through a different entrance of the building, or the user turns on the phone at some time once inside the building. The localization service will activate once the users pass through the first landmark.

In this chapter's implementation, we extracted the GPS location of the building entrance from Google Satellite View. During bootstrap, a user activated UnLoc when entering through this entrance. Alternatively, the entrance location could have been collected by other means, or even estimated from locations where GPS fixes are lost (after entering a building). We firmly believe that obtaining the GPS coordinate of one SLM, just one time, will not be difficult in real life.

Activity-based landmarks are feasible too – a busy cafe may invariably have a queue, or visiting a restroom may have a unique signature. These activities can very well be landmarks as long as their patterns surface upon clustering. Even temporary landmarks can be learned (i.e., queue exists between noon and 2:00 pm only), and even unlearned if the queueing behavior disappears during, say, winter vacations. Our current implementation has not explored these opportunities; we have only used signatures that are stable over time.

The early adopters of UnLoc will help with localizing the OLMs and bring the system to convergence – is this not war-driving? We argue that this is certainly not war-driving because the early adopters behave naturally and do not collect ground truth (since they do not need GPS). In fact, they collect exactly the same sensor readings as all other users. The only difference is that the early users may experience less localization accuracy. The process of war-driving, on the contrary, is associated with the notion of (ground truth) calibration, which naturally requires additional equipment.

The next section discusses the implementation, experiment methodology, and performance of UnLoc.

2.4 Evaluation

2.4.1 Prototype Implementation

UnLoc is implemented on Google NexusS phones using JAVA as the programming platform. The phone samples the four sensors (magnetometer, compass, and accelerometer at 24 Hz; gyroscope at highest permissible rate) and WiFi at 1 Hz. Various features derived from these measurements are sent to the UnLoc

server.[7] The server side code is written using C# and MATLAB, and implements the dead-reckoning, clustering, and landmark signature-matching algorithms. Whenever a new landmark is detected from clustering, the server updates the OLM list.

2.4.2 Methodology

We design real-life experiments with three different users in three different university buildings – (1) computer science (2) engineering, and (3) a shopping mall. Approximately, we covered 1750 m^2, 3000 m^2, and 4000 m^2 respectively, in these buildings. Each user walked around arbitrarily in the building for 1.5 hours, covering multiple floors; each carried two phones, one in the pocket and another in the hand with the screen facing up. We made separate arrangements to collect ground truth (recall that GPS is not available inside any of these buildings). Briefly, we pasted markers on the grounds at precisely known locations, such as the center of a classroom door, the first step in a staircase, the entry-point to the elevator, in front of a window, etc. Each of these markers had a number on it; as a user walked through a marker, the user spoke out the number on the marker, and the phone recorded it. By superimposing the map of the building on Google Earth, and identifying the corresponding locations of the markers, we extracted their GPS locations. This offered us ground truth at these markers. Between two markers (separate by 5 m on average), we interpolated using step count. Of course, UnLoc did not rely on any of the ground truth markers to compute its location. Thus, at any given time, the difference between ground truth and the UnLoc-estimated location, is UnLoc's instantaneous localization error.

2.4.3 Evaluation Results

We intend to concentrate on the following questions:

- How many landmarks are detected in different buildings? Are they well scattered? (Figure 2.16)

[7] In a real-life deployment of UnLoc, communication to the server may not be necessary. The landmarks of a building can be downloaded *a priori*; clustering and landmark matching algorithms can be executed locally on the phone.

- Do real users encounter these landmarks (i.e., is the matching between the sensor reading and established landmark signatures, reliable)? (Figure 2.17)
- What is the localization accuracy of a user, in real-time, and offline? (Figure 2.18, Figure 2.19, Figure 2.20)

2.4.3.1 SLM Detection Performance

Table 2.1 shows the confusion matrix for the detection of all SLMs (using traces from two malls in Egypt). The matrix shows that some SLMs are easier to detect than others due to their unique patterns. This leads to zero false positive and negative rates for the elevators and walking cases. However, even with the difficult SLMs, the UnLoc's template signatures can still achieve a high accuracy, with an overall 0.2% false positive and 1.1% false negative rates.

2.4.3.2 Detecting Organic Landmarks (OLMs)

Figure 2.16(a) shows the number of landmarks detected inside different buildings. For the engineering building, the breakup of the landmarks is: 9 magnetic, 8 turns, and 15 WiFi OLMs. For the computer science building, the breakup is: 9 magnetic, 10 turns, and 10 WiFi OLMs. Perhaps more importantly, Figure 2.16(b) shows how these landmarks are quite homogeneously scattered inside the buildings. We observe that with these numbers of well-scattered landmarks, a user's dead-reckoning error is not likely to grow excessively, in turn helping landmark localization. This well matches our core intuition and expectation.

Figure 2.17(a) reports the accuracy of the landmark locations, as computed by UnLoc. Observe that aligned with our intuition, the number of landmarks increase over time, as more users explore the space (Figure 2.17(b)). Moreover, the accuracy of these landmarks also increases, since different paths bring different independent estimates. In fact, our datasets are limited in diversity of paths since volunteers could not walk around into any rooms or auditoriums in these buildings – many were research offices, faculty offices, or classrooms. In reality, the diversity of different independent paths may be expected to augment the accuracy.

2.4.3.3 Landmark Signature Matching

If landmark signatures fluctuate quickly over time, then OLMs will be unstable. Thus, users may never encounter the established OLMs because their signatures are changing faster than they can be learned. Furthermore, it is entirely possible that users at a different location sense a signature that matches a faraway landmark. In such a case, the user's location will be repositioned to the (highly erroneous) landmark. To verify if such variations occur in our buildings, we collected sensor readings on multiple days. We found sound consistency in the signatures. This is not surprising because all our signatures are designed to be stable, particularly WiFi and accelerometer/gyroscope based turns. While the magnetic signatures can change, in our cases we did not observe anything appreciable.

Nonetheless, recall that UnLoc embraces a conservative approach by using a low "similarity threshold" while declaring a landmark. In other words, the signature of the landmark should be very dissimilar with other signatures to qualify as a landmark. This ensures that when a test user matches a sensed readings with existing OLMs, the false positive (FP) rate is low. Figure 2.17(c) quantifies false positives – evidently FP is less than 1%. As a tradeoff for choosing very distinct signatures, it is possible that a test user may not match it well. Figure 2.17(c) shows that the matching accuracy is reasonably high, although not perfect. We believe that this is an acceptable tradeoff – given that the number of landmarks are high, missing a few will affect performance much less than matching to an incorrect landmark. In other words, UnLoc is in favor of trading off matching accuracy to maintain low false positives.

Of course, changes in the ambience – say relocation of major electrical equipment to a different room, or deactivated WiFi APs – will affect existing landmarks, and UnLoc will not be able to match them. However, UnLoc will learn these changes over time, both the disappearance of the landmark from its original location, as well as the emergence of a landmark at a different location. So long as these changes are not all at the same time, UnLoc should be able to remain resilient to landmark churn. We leave the quantification of this claim to future work.

2.4.3.4 Localization Performance

In offline localization, whenever a user encounters a landmark, UnLoc learns the user's errors, and therefore can track back and partly correct the user's past trail. Online, real-time localization does not offer this benefit. Figure 2.18(a) shows the comparison of these two forms of localization – evidently, the advantage of offline is appreciable. This implies that for applications which do not need online tracking, localization error can be within 1.15 m on average, even with a few landmarks.

Beyond the bootstrapping phase, as UnLoc identifies several organic landmarks, error correction opportunities will increase. How much accuracy can we expect in steady state when many OLMs have already been recognized? Figure 2.18(b) demonstrates the result; mean instantaneous localization error is within 1.69 m. However, the performance will only improve over time as more OLMs are detected. Figure 2.19(a) quantifies the intuition. As a user moves away from a landmark, the location error grows and eventually gets reset at the next landmark. Initially, UnLoc only has a few landmarks and hence the error between two landmarks is high. As more landmarks are identified and added to the system, the error growth is curbed frequently. We aggregate the error over time across all the users walking on multiple routes, and present the findings in Figure 2.19(b). The benefits of additional OLMs are evident.

The above results are from experiments across three different buildings using several landmarks. It is difficult to predict how many such landmarks exist in other buildings. Thus, it is natural to ask if *localization performance will get derailed if only a few landmarks exist.* Figure 2.20 presents location error for varying landmarks. Even with 10 out of the 28 landmarks, average instantaneous location error is within 1.9 m. We expect a few landmarks to be available in most buildings; when in doubt, UnLoc could even turn on the microphone to expand to ambient acoustic signatures. While far more rigorous experimentation is necessary (across more buildings, people, phone platforms, and time), we believe the results from these small-scale tests are promising to justify moving forward.

2.4.3.5 Note on Performance Comparison

Table 2.2 shows a qualitative comparison of UnLoc with a number of other indoor localization schemes. The most relevant scheme for comparison is EZ [33], which to the best of our knowledge, was among the first to attempt calibration-free localization. However, we believe that EZ's reliance on "occasional GPS fixes" in indoor environments could be problematic. We believe UnLoc's requirement of only a door location is more dependable/scalable. Also, UnLoc is far less sensitive to RF signal fluctuations; the data volume needed to bootstrap is also less. Other schemes in Table 2.2 require war-driving or special infrastructure; UnLoc is free of both.

2.5 Limitations and Future Work

Heterogeneous Hardware It is possible that the landmark signatures will vary across smartphone platforms – UnLoc has not been tested for this. However, if data gathered from smartphones is indexed by the phone's make and model, it should be feasible to detect landmark signatures for each distinct model (as we have, for the Android Nexus series). Thus, in real life, a phone would download landmark signatures that are specific to its hardware, and run UnLoc for localization. In other words, UnLoc would extend to all platforms from which initial sensor data has been gathered.

Phone-Orientation Effect We have experimented with the phones in realistic orientation (in pocket and in hand). Handling arbitrary phone orientations and their effect on the reported sensor values is an important direction for future research. As gyroscopes are becoming more available on new phones, they can be leveraged to map arbitrary orientation to a specific frame of reference. Meanwhile, orientation-independent features, such as the magnitude of the acceleration, should be investigated.

Energy Footprint UnLoc avoids using sensors that have a high energy footprint, e.g., light and sound. This is expected to limit the energy consumed from sensing. Moreover, UnLoc can incrementally turn on sensors, perhaps depending on the density of landmarks available in the environment. For example, if a wing of a building has numerous landmarks, UnLoc could turn off the mag-

netometer, and use only the accelerometer and compass-based landmarks for localization. This chapter has not addressed such optimizations – these are part of our ongoing work.

Scalability Testing A complete UnLoc system needs to be tested over larger scale, in terms of the number of users and data size. However, as with other crowd-sourcing based systems, it is difficult for us to deploy a large-scale testbed. This includes in part providing incentive systems for users to deploy the system and experiment with it.

2.6 Related Work

Our goal and vision intersect with a number of projects in literature. In particular, the indoor localization literature is vast, ranging from theoretical models to simulations to implemented systems [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. In the interest of space, we heavily sub-sample this literature, focusing on systems related to UnLoc.

Calibration-Free Localization A recent work called *EZ* [33] was among the first to attempt indoor localization without war-driving. Their key intuition is that overheard WiFi APs and RSSI can together offer the user's location (via a genetic algorithm), provided that a mobile device occasionally gets a GPS fix. While EZ demonstrates accuracies between 2 m to 7 m, the GPS locks inside a building can be erratic. Moreover, the precise RF signal propagation may vary over time due to environmental dynamism. UnLoc eliminates this reliance on periodic GPS fixes, and only relies on a one-time global truth information e.g., the location of a door or staircase or elevator. The entire system can bootstrap from this.

WiFi-based Techniques Other RF based techniques [12, 14, 16] also inspire UnLoc. Place Lab [16] is a highly successful project where signals from different WiFi and GSM base stations are utilized for localization. A wireless map is created by war-driving a region; the mobile device localizes itself by comparing overheard APs/cell towers against the wireless map. UCSD's Active Campus project [17] adopts similar techniques of localization, but assumes that the lo-

cation of the WiFi access points are known *a priori*. RADAR [12] also operates on WiFi fingerprinting, and is capable of achieving up to 5 m accuracy in indoor settings. As a tradeoff to accuracy, RADAR needs to carefully calibrate WiFi signal strengths at many physical locations in the building. High-resolution calibration is time-consuming and may not scale over wide areas. Some techniques have bypasses the calibration effort through deployment of additional infrastructure, e.g., Cricket [21], Lease [15], PAL [19], Pinpoint [18]. UnLoc is designed to be an infrastructure-independent, calibration-free, system.

Dead-Reckoning Dead-reckoning using inertial sensors is a well-studied research area. However, typical sensors used in such domains are expensive and high-quality ones; coping with noisy smartphone sensors in indoor environments is far less explored. The key problem is that dead-reckoning suffers from the accumulation of error, and can grow cubically even with foot-mounted accelerometers [34]. To reduce such error, periodic recalibration with the GPS has been used in outdoor environments [13, 35]. However, GPS is not available indoors. UnLoc presents an option to replace GPS with indoor landmarks; the ability to identify these landmarks in an unsupervised way enables zero-calibration indoor localization.

Simultaneous Localization and Mapping (SLAM) A highly popular and successful technique in robotics, called SLAM, allows a robot to simultaneously discover landmarks and build a map-representation of an indoor environment [36]. However, SLAM typically depends on using explicit environment sensors, such as laser range finders and cameras. Moreover, the rotation of the robot wheels offer a precise computation of displacement. Recently, WiFi-SLAM [37] proposed the usage of the WiFi signal strength for SLAM based on a Gaussian Process Latent Variable Model (GP-LVM). However, the multi-path propagation affects the accuracy of the model that captures the relation between the WiFi signal and distance. The chapter also assumes that motion between multiple WiFi samples can be accurately determined. UnLoc is certainly reminiscent of SLAM, even inspired – however, it is completely different. Unlike SLAM, UnLoc uses smartphone sensors to compute the displacement and direction of users; the landmarks are essentially ambient signatures or user-activities. While ambient signatures and activity recognition have been

utilized in the past [25, 38], their applicability toward dead-reckoning is, to the best of our knowledge, novel.

Sensor Fusion Multi-modal sensing has received strong interests – [39] uses a body worn sensory board (MSB) for human activity recognition. UnLoc, however, is an early attempt to apply activity recognition to indoor location tracking, using cheap mobile phone sensors.

2.7 Conclusion

We observe that inherent properties of indoor environments offer unique opportunities to perform localization. The core approach draws from existing ideas in the literature, and combines them in an unconventional way. Essentially, mobile devices are dead-reckoned using their sensor measurements, but these same measurements are leveraged to detect unique environmental signatures within the building. These signatures are used to correct the dead-reckoning error, which in turn improves the location accuracy of these signatures – a recursive procedure. Performance results suggest promise, motivating us to pursue UnLoc to the point of real deployment.

2.8 Figures

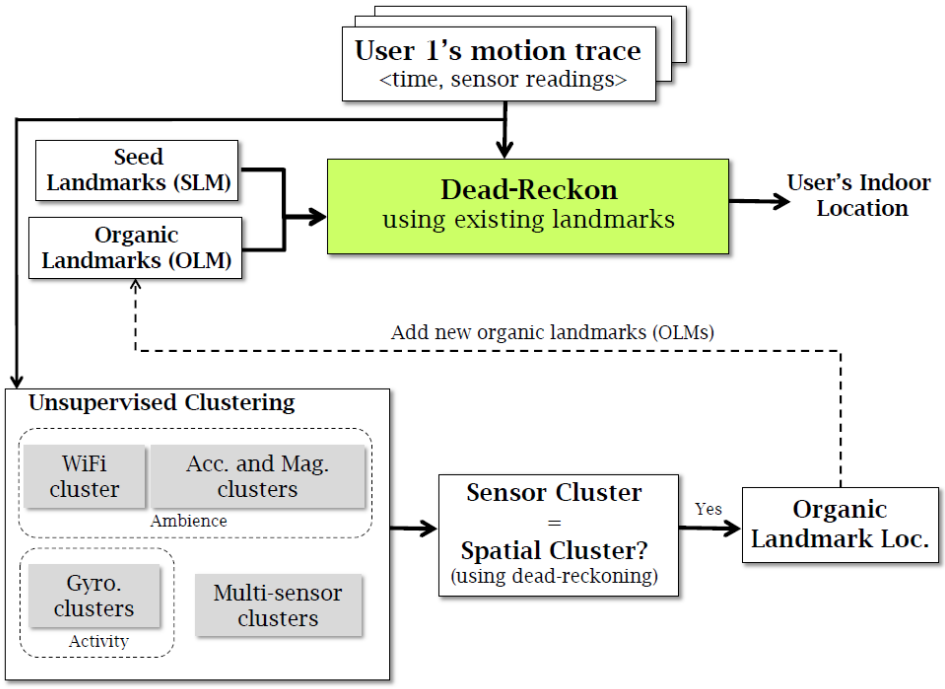


Figure 2.1: UnLoc architecture – motion vectors are computed from recorded sensor readings, and the user's location computed via dead-reckoning. The dead-reckoning error is periodically reset using landmarks in the environment. Further, the same sensor readings are mined to identify signatures of new landmarks. These landmarks help in improving localization accuracy for subsequent users.

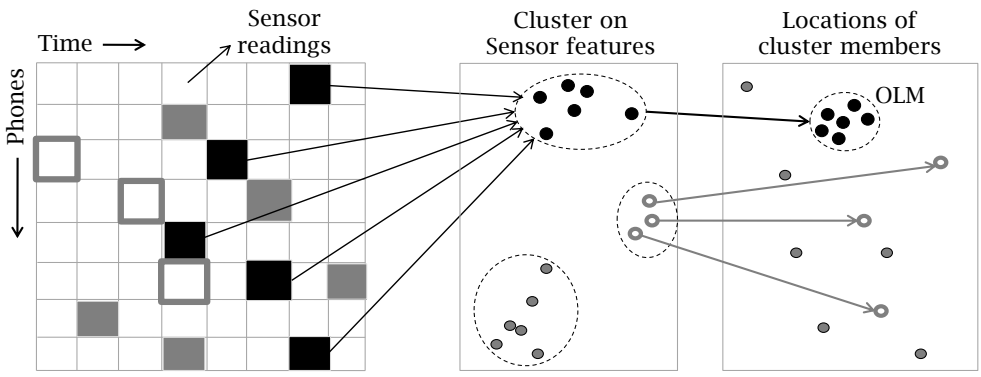


Figure 2.2: Matrix showing sensor readings collected by devices across time. Readings are clustered and location of cluster members computed. If all cluster members fall within a small region, UnLoc deems it an OLM.

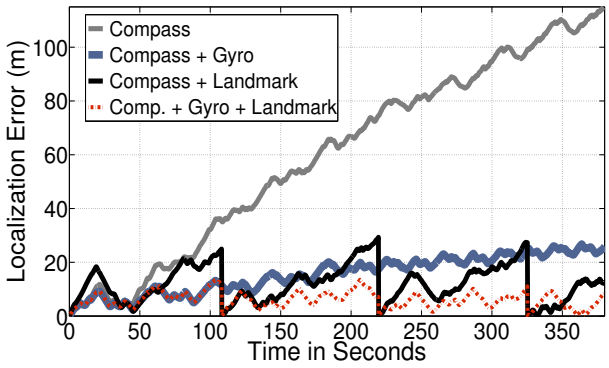


Figure 2.3: Error from dead-reckoning reduces when using a gyroscope, and further reduces with landmarks.

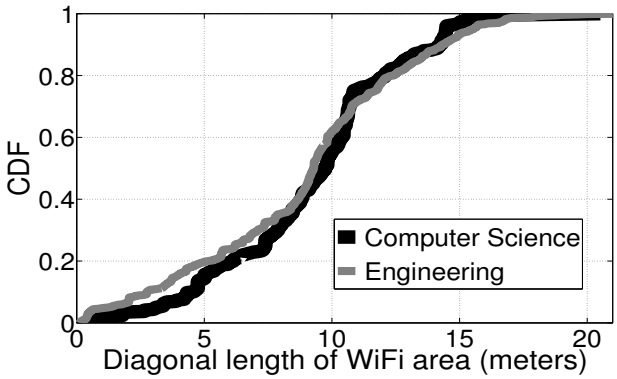


Figure 2.4: Some WiFi areas are very small (tail of distribution), and hence, an ideal landmark.

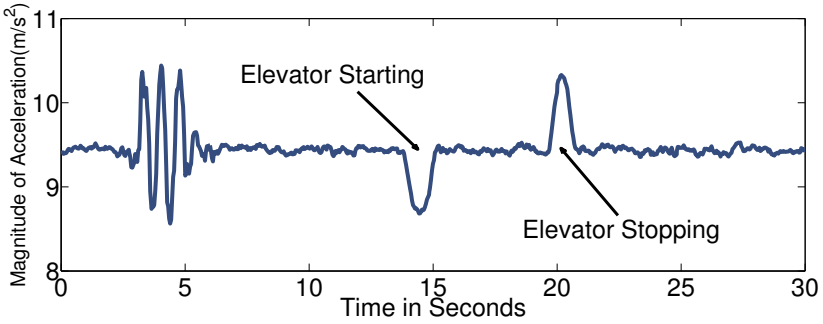


Figure 2.5: Accelerometer signature inside the elevator (caused by the elevator starting and stopping).

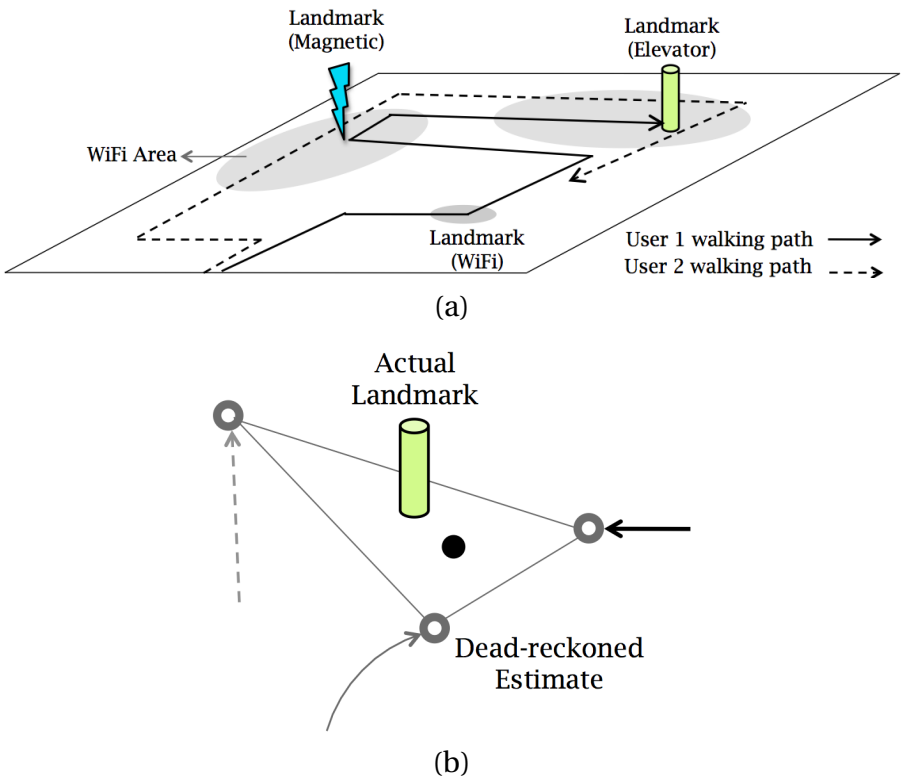


Figure 2.6: (a) UnLoc users walk and periodically encounter landmarks – refine landmark locations, correct own locations. (b) The solid circle shows the centroid of the dead-reckoned estimates. Multiple erroneous estimates lead to a better approximation of the landmark location.

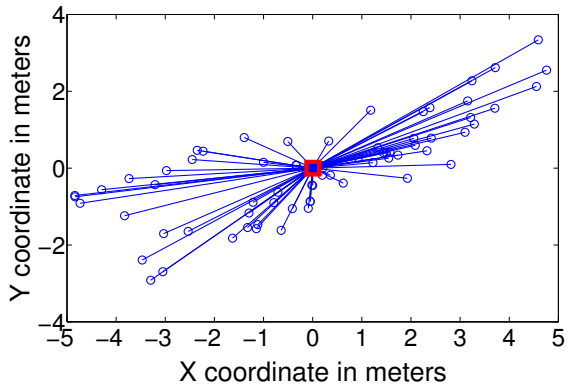


Figure 2.7: Uncorrelated errors from multiple dead-reckoning estimates.

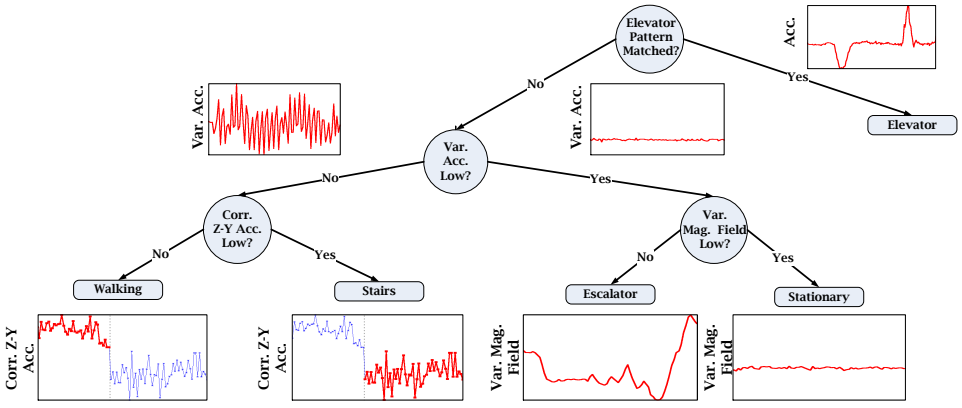


Figure 2.8: Decision tree for detecting Seed LandMarks (SLMs). The top level separates the elevator based on its unique acceleration pattern. The second level separates the constant velocity classes (stairs and escalator) from the other two classes (walking and stairs) based on the variance of the acceleration. The third level uses the variance of magnetic field to separate the escalator from the stationary case and the correlation between the Z and Y acceleration components to separate between the stairs and walking cases.

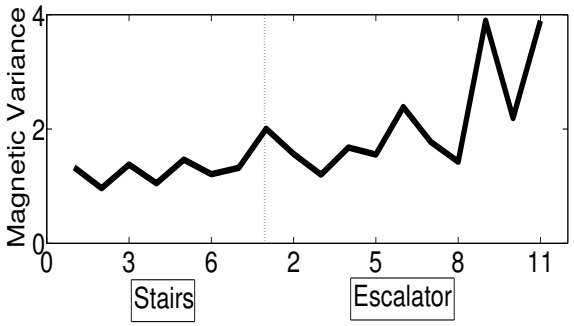


Figure 2.9: Magnetic variance when a user is climbing stairs versus using an escalator.

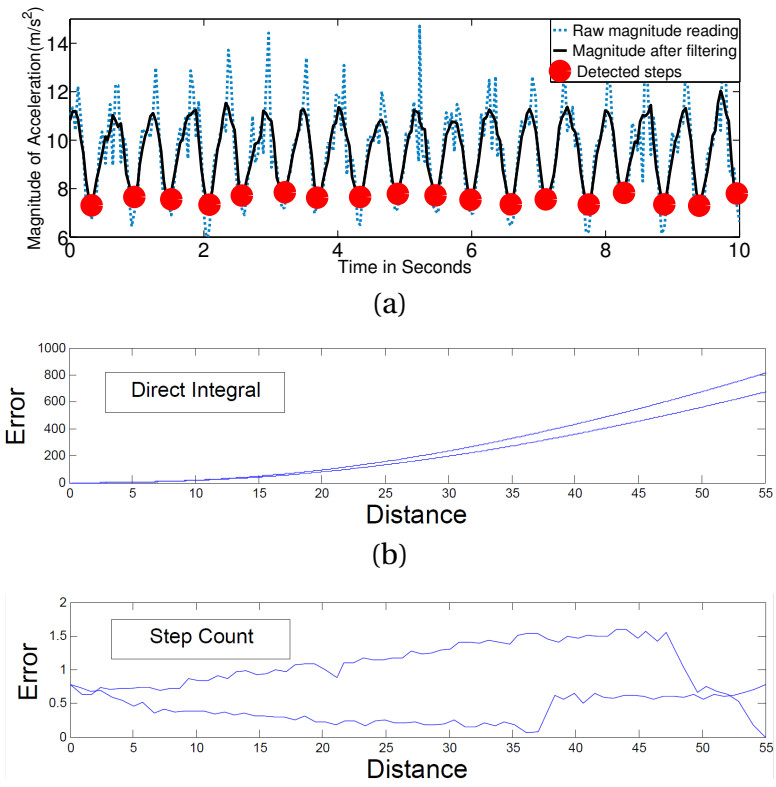
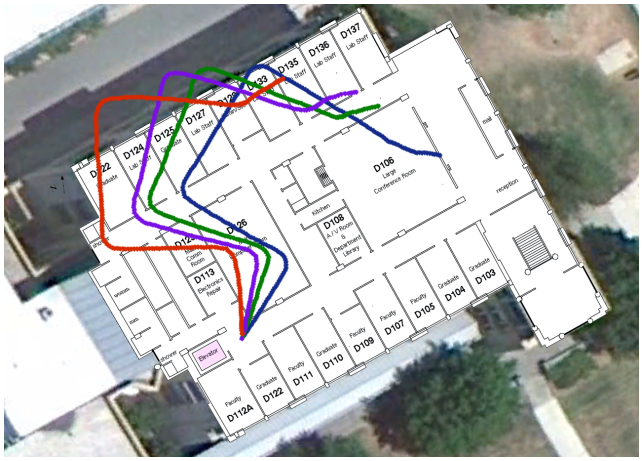
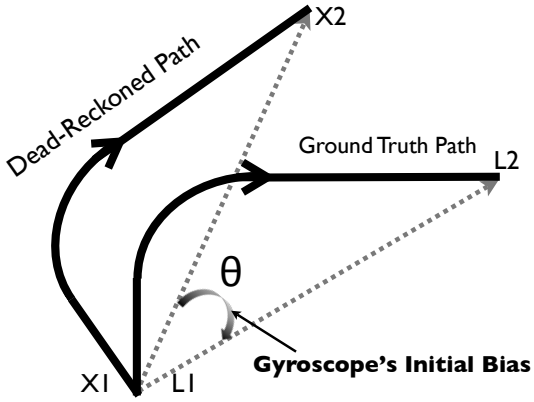


Figure 2.10: (a) Accelerometer readings (smoothed) from a walking user. (b) Displacement error with double integration for two users. (c) Error using the step-count method.

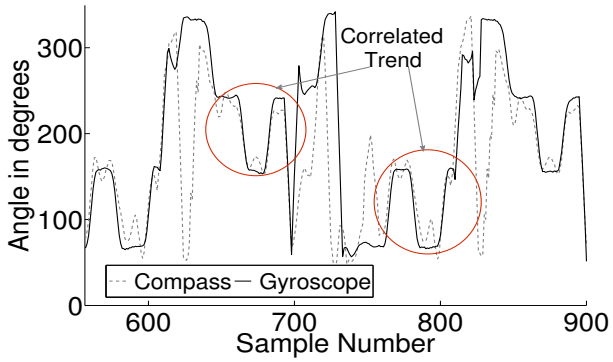


(a)

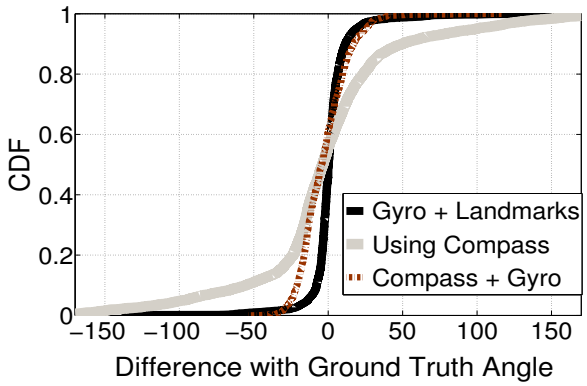


(b)

Figure 2.11: (a) Rotated walking trail due to the gyroscope's initial bias. (b) Correcting the gyroscope's bias using landmarks.



(a)



(b)

Figure 2.12: (a) Compass orientation is more reliable when it is correlated with the gyroscope. (b) CDF of orientation estimation error.

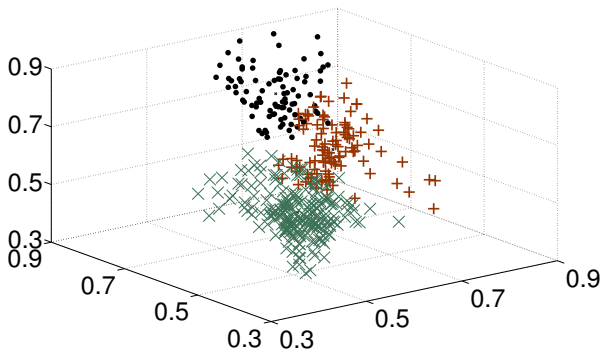


Figure 2.13: Clusters identified by K-means algorithm.

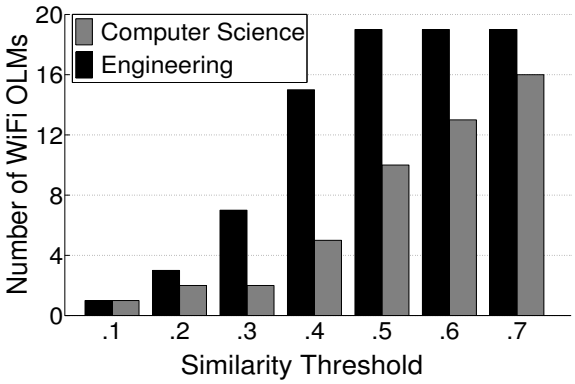


Figure 2.14: Tradeoff between similarity threshold and number of WiFi landmarks.

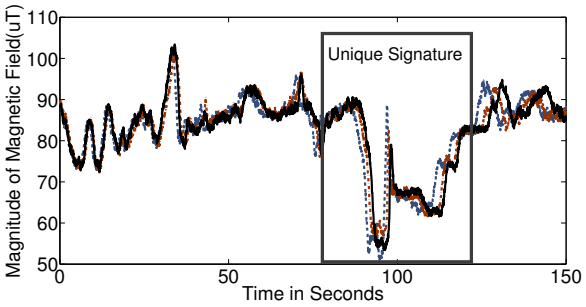
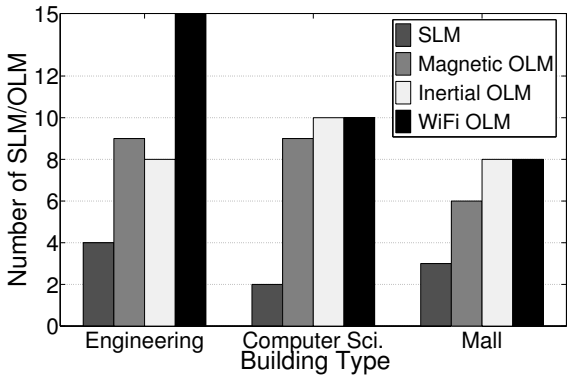


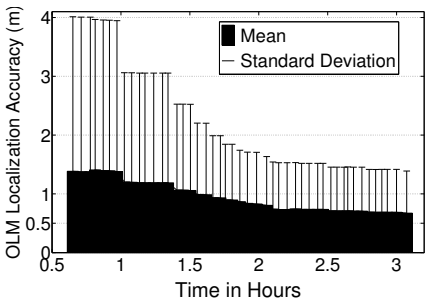
Figure 2.15: Magnetic signature near networking lab.



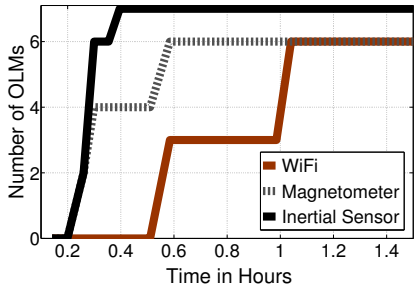
(a)

(b)

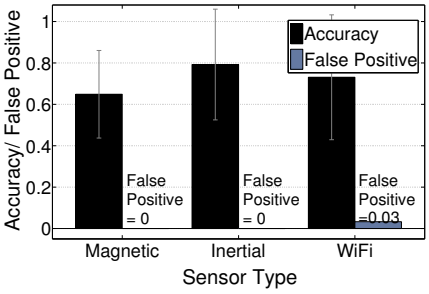
Figure 2.16: (a) Number of SLMs and OLMs located in different buildings. (b) Location of different types of OLMs in the engineering building.



(a)

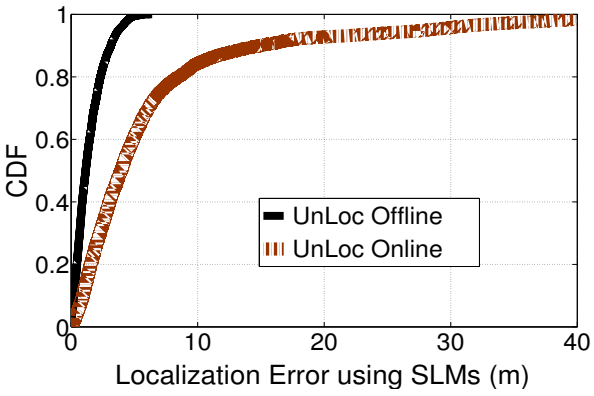


(b)

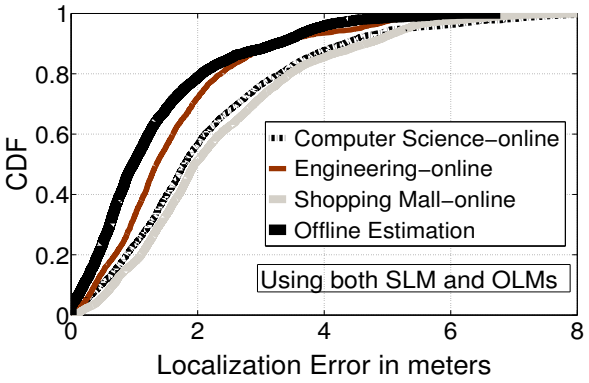


(c)

Figure 2.17: (a) OLM localization accuracy improves over time. (b) More OLMs are detected as more users walk around the building. (c) OLM matching accuracy and false positives.

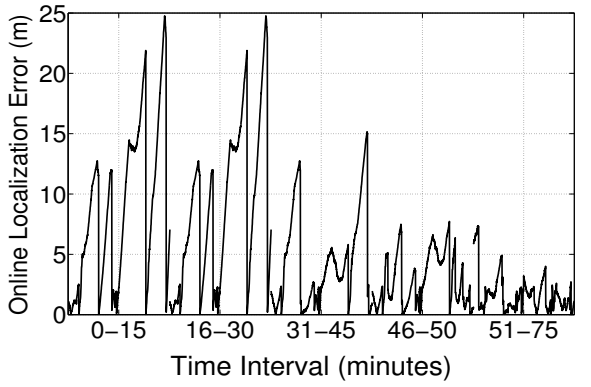


(a)

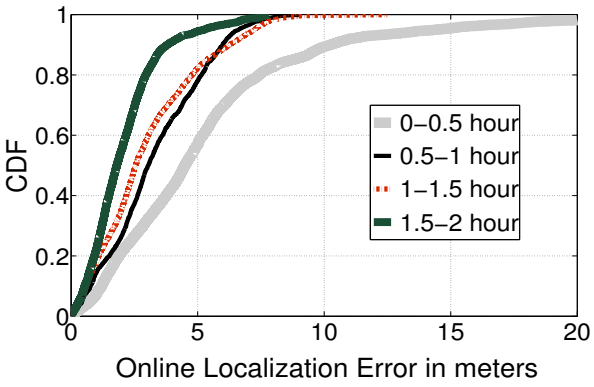


(b)

Figure 2.18: (a) CDF of localization accuracy using only SLMs. (b) CDF of localization accuracy using both SLMs and OLMs.



(a)



(b)

Figure 2.19: (a) Localization error over time for a single user. (b) Localization error over time averaged over all users.

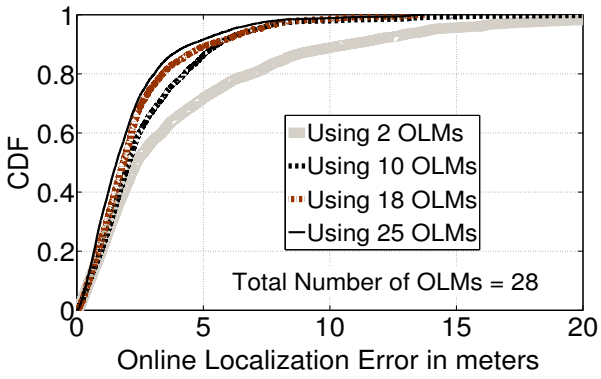


Figure 2.20: Tradeoff between number of landmarks and localization performance.

2.9 Tables

Table 2.1: Confusion matrix for classifying different seed landmarks.

	Elevator	**Stationary**	**Escalator**	**Walking**	**Stairs**	**FP**	**FN**	**Traces**
Elevator	24	0	0	0	0	0%	0%	24
Stationary	0	31	1	0	0	0%	3.1%	32
Escalator	0	0	22	0	0	0.6%	0%	22
Walking	0	0	0	39	0	0%	0%	39
Stairs	0	0	0	1	52	0%	1.8%	53
Overall						0.2%	1.1%	170

Table 2.2: Comparison with other localization systems.

Name	**EZ**	**SLAM**	**Horus**	**UnLoc**
Accuracy	2 to 7 m	~ 5 m	~ 1 m	1 to 2 m
Pitfalls	GPS Lock?	Special Sensors (LIDAR, Odometer.)	RF Sensitivity	Door Location
Overhead	None	None	War-driving	None

CHAPTER 3

VIDEOLOC: VIDEO BASED INDOOR LOCALIZATION

In Chapter 2, we discussed UnLoc, which achieves high median accuracy of 1.69 m without infrastructure and calibration costs; however, the tail of the error distribution is long. In some applications, 99th percentile accuracy should be high and people are willing to pay the price for such accuracy. Therefore, we also studied solutions using some infrastructures, VideoLoc, where feeds from surveillance cameras can be leveraged for highly precise localization, without compromising the privacy of individual users.

While computer vision technologies can estimate a user's location from surveillance cameras, the key question in VideoLoc is how to send the estimated location from the camera to the user's smartphone. To enable such communication, we need a visual address for the user. While the user's face could be one possible approach, however, we believe that it may not always be visible. Our core technique [8, 9] exploits the intuition that human motion patterns and clothing colors can together encode several bits of information. Treating this information as a "visual address", it is feasible to enable communication between a camera and a user, while allowing the user to turn off the "visual address" at will. This core technique in VideoLoc can be extended to other applications such as augmented reality. Therefore, in this chapter, we will focus on this core technique and discuss it in a general manner. Results from real-world experiments show that 12 individuals can be discriminated with 90% accuracy using six seconds of video/motion observations. Video-based emulation confirms scalability up to 40 users. At the end of the chapter, we evaluate the localization accuracy of VideoLoc. Results from six users in our engineering building demonstrate median location errors of 0.11 m and 99th percentile errors of 0.58 m.

3.1 Visually Fingerprinting Humans without Face Recognition

This section develops techniques by which humans can be visually recognized. While face recognition would be one approach to this problem, we believe that it may not always be possible to see a person's face. Our technique is complementary to face recognition, and exploits the intuition that human motion patterns and clothing colors can together encode several bits of information. Treating this information as a "temporary fingerprint", it may be feasible to recognize an individual with reasonable consistency, while allowing the user to *turn off* the fingerprint at will.

One application of visual fingerprints relates to augmented reality, in which an individual looks at other people through a camera-enabled glass (e.g., Google Glass) and views information about them. Another application is in privacy-preserving pictures – Alice should be able to broadcast her "temporary fingerprint" to all cameras in the vicinity along with a privacy preference, saying "remove me". If a stranger's video happens to include Alice, the device can recognize her fingerprint in the video and erase her completely. This section develops the core visual fingerprinting engine – *InSight* – on the platform of Android smartphones and a backend server running MATLAB and OpenCV. This section is published in MobiSys 2015 [9].

3.1.1 Introduction

Imagine a near future where humans are carrying smartphones and wearing camera-embedded glasses, such as the Google Glass. This section intends to recognize a human by looking at the individual from any angle, even when the face is not visible. For instance, Alice may look at people around her in a social gathering and see the names of each individual – like a virtual name tag – suitably overlaid on her Google Glass display. Where revealing names is undesirable, only a short message could be posted. People at the airport could post "looking to share a cab", students in a startup event could post "seeking a co-founder", and Alice could view each individual's posts above their heads (Figure 3.1). In general, the ability to differentiate individuals visually could enable human-centric augmented reality [40, 41].

Face recognition [42, 43] is a possible approach to the above problem. However, faces are not always visible. Moreover, many people express discomfort releasing their profile pictures to the cloud, given that it can become a permanent identifier for de-anonymizing other content in the web [44]. Ideally, what is necessary is a *temporary* visual identifier, that can be activated at will, and will identify the individual momentarily but not later.

This section pursues the intuition that human motion patterns and visual appearance (e.g., clothing colors) can together serve as a temporary visual fingerprint. The key idea is simple. Consider Alice looking at an individual X through a Google Glass (or smartphone camera). The *InSight* server could request Alice to upload a short video snippet of that individual, and use the frame sequence in this video to extract a motion fingerprint of X, denoted by V_X^{Alice}. This motion fingerprint is essentially a string of micro-activities such as walking direction, stepping frequency, stopping, turning, etc., extracted from the video. The server can simultaneously request sensor data (e.g., accelerometer, gyroscope, compass) from people around Alice, and extract a similar motion fingerprint from it. Let M_i denote this sensor-based motion fingerprint for user i. By matching V_X^{Alice} against M_i of each user i, the server can find the strongest match, say for $i = Bob$. The server can convey to Alice that she is looking at Bob and display Bob's message (e.g., "looking for interns") on her glass.

Generalizing, visual fingerprints may not only be from motion patterns, but also from clothing colors, body structure, etc. If Alice recognizes Bob through motion fingerprints, she can extract Bob's clothing features and update a database inside the *InSight* server. In the steady state, the database would cache clothing fingerprints for different individuals. When John looks at Bob later, his Glass only needs to send an image of Bob. The server can extract the clothing fingerprint from the image sent by John and match against precomputed clothing fingerprints, ultimately notifying John that he is looking at Bob. In summary, we believe that a person's non-facial visual appearance can serve as an identifier. There is evidence of this opportunity given that humans can often recognize other humans without looking at their faces. This section demonstrates that (wearable) cameras and smartphones can together achieve the same.

Realizing the above idea presents a number of challenges. Extracting fingerprints from sensors and videos can be non-trivial, even though a variety of tools are available in the signal processing and computer vision literature. The fingerprints need to be general for scalability across individuals, while being adequately discriminating for identification. Moreover, fingerprint matching must be done across incompatible dimensions (sensor and vision) requiring the system to cope with normalization issues, dynamic ranges, depth, perspectives, etc. Even for matching clothing fingerprints, challenges emerge due to lighting conditions, wrinkles, and various view angles – the front and back of a dress may have different colors and patterns. Finally, the system needs to support incremental deployment (i.e., not everyone may run *InSight*) while bandwidth and energy overheads should be minimal.

While developing a robust system is challenging, we find that the rich diversity in human behavior offers promise. People walk/turn/pause at different time instants, even when they are walking in groups – observed long enough, each individual's motion sequence should begin to become unique. Encouraged by this opportunity of uniqueness, we adopt a "digital" approach to processing the information. Put differently, we express fingerprints as *strings* defined on a pre-specified motion alphabet. An example fingerprint could be $EEEOOR$..., where E, O, and R correspond to the actions of *walkEast*, *noMotion*, and *turnAround*, respectively. Such motion alphabets are extracted from both sensors and videos, allowing *InSight* to employ string matching algorithms for comparing fingerprints.

InSight translates these ideas into a functional system using Android Galaxy phones and videos taken from Google Glasses. We have not attained real-time operations yet – the server runs on MATLAB with links to OpenCV and machine learning libraries, and returns the result within ten seconds. Real-world evaluations demonstrate the ability to discriminate 12 individuals with 90% accuracy, using six seconds of video/motion observations. Video-based emulation shows the ability to scale the technique to the order of 40 people. The main contributions may be summarized below.

- *Identifying the possibility that human clothing colors and motion patterns could serve as temporary fingerprints, complementing face recognition.* We use these fingerprints as new degrees of freedom for human-centric visual applications.

- *Quantifying the viability and accuracy of fingerprinting with real-world human behavior.* We build a fully functional prototype and demonstrate promise through micro-benchmarks, real-user evaluation, and larger-scale video simulation.

The subsequent subsections will expand on these contributions beginning with an overview of *InSight*, followed by a detailed system design. However, we first discuss a few potential applications.

3.1.2 Applications

The goal of this section is to develop the core visual fingerprinting primitives, with the goal to enable new use-cases or aid known applications. We briefly discuss a few possibilities here different from the augmented reality application described above.

3.1.2.1 Privacy Preserving Pictures/Videos (PPP)

The proliferation of cameras, and wearables cameras in recent years, has raised various discussions on privacy. Many citizens have expressed discomfort at the thought of being included in a video or a picture taken by a stranger, even if it was legally done in a public place. Today's solution is to hastily move out of the camera's field of view when it is clear that a picture is being taken. Of course, this is not always possible because people are often unaware that videos or pictures are being taken around them. Sensitized by this, many privacy conscious users have wished for a capability to express their privacy preferences, such as "please remove me from the video".

Now, assume Alice is taking a video, and Bob, present in the field of view, intends to be removed from Alice's video. Bob can share his motion fingerprint with the server. When Alice takes the video and sends it to the server, visual fingerprints can be computed for every individual in the video, and compared against Bob's. A match suggests Bob is indeed in the video – *InSight* can then remove Bob by replacing him with background imagery in the video. Authors in [45] recently proposed using QR codes on clothing as a means of expressing privacy preferences. We believe *InSight* is a more usable solution.

3.1.2.2 Visual Addressing and Communication

A grocery store is in conversation with us regarding the applicability of *InSight* for customer localization and communication. The idea is to use wide-angle surveillance cameras mounted on high ceilings to observe the top view of moving customers (each customer visible to the camera as a small blob). Consider the case where Alice is shopping in the store and her smartphone records her motion fingerprint, M_{Alice}. The surveillance camera could also compute her motion fingerprint from the video frames, V_{Alice}. Of course, it might take longer since the motion alphabet (from a top view) will be limited — the camera would only detect moving, paused, and moving direction. However, even these few bits of information should be adequate to disambiguate customers in the scale of a minute (no two customers move/pause in lock step for that long).

Now, it should be possible for the camera to send a message to Alice, by including V_{Alice} as an address inside the message (like a virtual MAC address). When Alice's device receives the message, a comparison between M_{Alice} and V_{Alice} would indicate that the message is meant for her. Thus, the grocery store can now establish a communication channel with Alice, sending her location-based product information (since the camera knows Alice's exact location). Even Alice can ask questions such as "where can I find brown rice?" and the store can respond with "ahead on your right, at the bottom shelf". Observe that this *visual address-based communication* offers privacy since Alice need not reveal her (permanent) MAC/device IDs. Moreover, she can turn off her motion sensors at will, terminating all interaction with the store.

While the above applications may not be the best, they define the design landscape reasonably well for us to build a generic visual fingerprinting system. The most suitable applications, we hope, will emerge in time.

3.1.3 System Overview

This subsection presents a functional overview of *InSight*; the technical components will follow in Section 3.1.4. For preserving context, we use the augmented reality application as the central theme in the rest of the section, although the techniques extend (with minor modifications) to the other applications.

Consider a university-organized matchmaking event for startups, where a variety of students and faculty come with the goal of forming teams. Upon entering, users check-in with the *InSight* server and specify their rough locations. Later, say Alice looks at an individual X through her Google Glass (or smartphone camera) and requests to identify the individual. During the initial bootstrap phase, the *InSight* server running in the cloud asks Alice's glass to record and upload a video snippet of that individual. The server also requests sensor data (accelerometer, gyroscope, and compass) from smartphones of all people around Alice (Figure 3.2). Once the data arrives, the server processes these sensor data and extracts motion fingerprints, M_i, for each user i. This motion fingerprint is essentially a feature vector, where features include isStanding, isWalking, walking–direction, step–duration, phase, isRotating, pause–timings, etc. The server also analyzes the video snippet from Alice and computes a similar motion fingerprint, but from the consecutive frames of the video. Let V_X^{Alice} denote this video-based motion fingerprint. By matching V_X^{Alice} against all values of M_i, *InSight* finds the strongest match, say for $i = Bob$. If this matching score is greater than a confidence threshold, the server conveys to Alice that she is looking at Bob, and displays the message Bob intends to share with others, e.g., "PhD student in CS, looking for CEO for mobile analytics startup".

Now, once *InSight* recognizes Bob in the video snippet, it extracts Bob's color fingerprint, C_{Bob}, and updates a fingerprint database. This fingerprint essentially captures color features and patterns from Bob's clothing. Thus, in the steady state, when everyone's color fingerprint is registered in the database, users no longer need to update videos or sensor readings. If John looks at Bob later (see Figure 3.3), his Glass only needs to send to the server an image of Bob. The server extracts the color fingerprint from John's image, C_X^{John} and matches against all C_i in the database (the candidate set can be trimmed using John's rough location). Assuming C_X^{John} matches best with $C_{i=Bob}$, and the matching score is above a threshold, the server informs John that he is looking at Bob. Of course, these color fingerprints are valid in the time scale of events – for another event on the next day, people will be wearing different clothing, and the color database will have to be re-populated.

We make two observations about the properties of motion and color fingerprinting.

1. A short window of sensor data is mostly adequate to compute a discriminating motion fingerprint for an individual. This is due to the inherent diversity of human motion, i.e., people's micro-motions are not likely to be synchronous for long durations, and the first instance of "asynchrony" can be used to tell them apart.

2. As mentioned earlier, once Alice identifies Bob using motion fingerprints, Bob's color fingerprint, C_{Bob}^{Alice}, is added to the fingerprint database ($C_{Bob} = C_{Bob}^{Alice}$). When John identifies Bob later, perhaps from a different angle, Bob's fingerprint is further refined ($C_{Bob} = C_{Bob} \cup C_{Bob}^{John}$).

In general, people's color fingerprints increasingly become complete over time, which improves the accuracy of recognition, which further completes the fingerprint. Thus, *motion fingerprints are only necessary to "register" a person for the first time. Once* InSight *has bootstrapped, color profiles become effective, and motion fingerprints are used only to boost matching confidence, if necessary.* The details on what constitutes color and motion fingerprints are presented in Section 3.1.4.

3.1.3.1 Matching Motion Fingerprints

Motion–fingerprint matching at the *InSight* server is non-trivial because the fingerprints are in different domains – M_{Bob} is obtained from accelerometer/gyroscope/compass readings, while V_X^{Alice} is extracted from video frames. To bring compatibility, we propose to translate all motion fingerprints into a common *semantic alphabet*, where example alphabets are "walking north", "rotating", "pausing" (see example alphabets in Table 3.1). Thus, both M_{Bob} and V_X^{Alice} are represented as strings on this alphabet, and fingerprint matching boils down to string matching. As an example, say Bob and Neil walk northward for three time units, and then Bob pauses while Neil continues walking for one more unit, and then Neil pauses too. Their respective strings will then be $M_{Bob} = NNNOO...$ and $M_{Neil} = NNNNO...$, and it would be possible to tell them apart at the fourth time unit. Specifically, if Alice is looking at Bob, then *InSight* will compute $V_X^{Alice} = NNNOO...$, which is expected to match better with M_{Bob}. In our actual implementation, the motion alphabet is far more sophisticated, including different directions of walking, duration and phase of walking steps, rotations, etc. In fact, for the above case, *InSight* will analyze

the duration and phase of walking steps, and unless Bob and Neil are well synchronized in their footsteps, they will be separated within four time units.

3.1.4 System Design

This subsection describes the extraction of motion and color fingerprints from sensor and video data, followed by fingerprint matching schemes. The techniques borrow from literature where suitable, with appropriate adaptions to this specific cross-sensor application.

3.1.4.1 Extracting Motion from Sensor Data

We begin with a description of how sensor readings from Bob's smartphone are translated into a motion string. The key motion alphabets that make up this string are derived from: (1) rotation, (2) walking, (3) walking step duration, (4) walking step phase, and (5) walking direction.

Rotation Detection Since the phone can be in an unknown orientation, we cannot rely on any single axis of the gyroscope (x, y, or z) to properly detect rotation. Therefore, we first project the rotation rate vector $r = (r_x, r_y, r_z)$ on to gravity vector $g = (g_x, g_y, g_z)$ in the phone's coordinate system, i.e., $r_{gravity} = (r_x g_x + r_y g_y + r_z g_z)/\sqrt{g_x^2 + g_y^2 + g_z^2}$. Then, the rotation angle around gravity is $\alpha_{gravity} = \int r_{gravity} dt$. If $\alpha_{gravity}$ exceeds a certain threshold, the user's motion is labeled as rotating (denoted by R in the motion alphabet). The alphabet R is essentially a binary indicator and its natural to ask: *why not extract more bits of information from rotation?* This is because detecting various degrees of human rotation from videos is a difficult problem.

Walking Detection The act of human walking manifests on the accelerometer with periodic high/low impulses as well as some rotation around the gravity axis. However, due to movements of the phone inside the pocket, and certain unusual gait patterns, simple threshold-based schemes were inadequate to recognize walking patterns. Instead, we employ a bagged decision tree model and train it with two features, namely (1) standard deviation on the magnitude of

accelerometers, and (2) rotation around gravity. We omit details in the interest of space, but found this to offer high consistency (as evident in Section 3.1.5.1).

Walking Step Duration Given that people walk with varying speeds, the duration of walking steps can be a useful discriminator. To this end, we first detect if the user is walking; if so, we apply a standard Kalman filter on the accelerometer data to first smooth out the reading and accurately identify the local maxima. These local peaks shown in Figure 3.4 are actually the time points when the human feet land on the ground. The peaks closer to the taller raw impulses correspond to the leg that carries the phone. The step duration, denoted T_{step}, may be computed as half of the time window between two consecutive peaks. Note that T_{step} can be computed without prior knowledge of which pocket the phone is in. Even if the phone is in the shirt pocket, or held in the hand, such peaks are visible, and T_{step} can be computed. We have not evaluated the case of backpacks and jacket pockets.

Walking Phase Even when two users are walking at the same speed, i.e., T_{step} is identical, their exact step timings may be out of phase. Thus, the phase of walking can also be a component of the motion fingerprint of a person. To this end, we use the peak around the bigger jerk as the step phase marker, shown in Figure 3.4. If different users exhibit these peaks at different times, they may be separated so long as time is appropriately synchronized among devices. If the accuracy of synchronization is upper bounded by, say δ, then the phase differences can be measured in that granularity. We show later that our synchronization is in the order of $\frac{T_{step}}{3}$, permitting us to create three buckets of users with respect to their walking phases.

Walking Direction *InSight* intends to leverage the user's walking direction, with some granularity, as an attribute of her motion. However, walking direction estimation is challenging given that the phone is in an unknown orientation on the user's body. The problem is difficult because the act of walking imposes various kinds of vertical, horizontal, and sideward forces on the smartphone (to stabilize the body), and there is a narrow window during which the acceleration on the phone is most dominantly along the user's heading direction. This narrow time window is actually when the user's leg swings (or rotates) forward, captured by the cross product of rotation axis, $\mathscr{R}_{axis}$ and gravity g.

More precisely, the user's heading vector, $H = \mathscr{R}_{axis} \times g$, illustrated in Figure 3.5. Given the rotation matrix $\mathscr{R}$ from the gyroscope, $\mathscr{R}_{axis}$ is the null space of $(\mathscr{R} - I)$ (I is the identity matrix) and rotation angle $\mathscr{R}_\theta = arccos(\frac{trace(\mathscr{R})-1}{2})$. In our system, we make $\mathscr{R}_\theta$ always positive and the sign of $\mathscr{R}_{axis}$ is determined by the right-hand rule. When the leg carrying the phone (called the primary leg) swings, the $\mathscr{R}_{axis} \times g$ points to user's heading direction; the same cross product points in the opposite direction when the secondary leg swings.

Importantly, this cross product must be computed when the leg is in full swing, otherwise, the sensor data can be polluted with noise (especially when the leg slows down and strikes the ground). To avoid such noise, *InSight* chooses a period of 30% of T_{step} starting from the time when the secondary leg strikes the ground (i.e., when the primary leg is about to swing). $\mathscr{R}_{axis}$ is derived from this time window. Figure 3.6(b) shows the $H(t) = \mathscr{R}_{axis}(t) \times g(t)$, where $\mathscr{R}_{axis}(t)$ is the rotation axis during $[t, t + 30\%T_{step}]$. Clearly, $H(t)$ alternates its direction as the user swings its primary and secondary legs alternately.

The heading vector is in the phone's coordinate system and needs to be interpreted in the global magnetic reference frame (i.e., with respect to North). For this, we project the magnetometer reading to the horizontal plane [46], and record the angle between this projected vector and the heading vector. Figure 3.6 shows the results. Vertical dotted lines are the moments when the heading vector is recorded. As expected, heading direction alternates roughly 180° between primary and secondary steps. At the moments of heading vector recordings, the direction matches the ground truth as it corresponds to the window when the primary leg swings. Upon close scrutiny, we find that the estimated walking directions are always slightly higher than the ground truth. This is because human leg motions are often slightly diagonal, and the actual walking direction is an average of two diagonals from two steps, resulting in forward locomotion. We use a calibration factor of 35° to compensate for this effect. The final walking direction is divided into eight directions, in the granularity of 45°, which creates eight motion alphabets. We describe how the same motion alphabets are also derived from videos (of Bob) captured from (Alice's) Google Glass camera earlier.

3.1.4.2 Extracting Motion from Video

Given that a video can contain multiple moving individuals, *InSight*'s first step is to mark and track every person in the video. This calls for enveloping each individual with a bounding box. Of course, for motion related insights (such as walking period, phase, etc.) lower regions of the bounding box need to be processed to extract alphabets. This subsection describes each of these steps systematically.

Detection and Tracking To detect and track humans in the video, we borrow existing techniques from computer vision [47, 48] and modify it per *InSight*'s needs. Using the borrowed techniques, Figures 3.7 (a)-(e) show the accuracy of placing a bounding box around each person, along with a confidence score. To cope with false positives, we only consider the boxes with a confidence score above 50. Now, to track people across the video frames, we again employ a Kalman filter [49]. Formally, the state of each *tracker* is denoted by $\mathbf{s}_k = [x_k, y_k, v_{x_k}, v_{y_k}]^T$, where $\{x_k, y_k\}$ and $\{v_{x_k}, v_{y_k}\}$ are the position and speed of the person in the 2D frame k. Observation z is the center of the bounding box obtained from the pedestrian detector. We use the following state and observation equations.

$$s_{k+1} = Fs_k + \omega_k \tag{3.1}$$

$$z_k = Hs_k + u_k \tag{3.2}$$

where

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

dt in F is the duration between adjacent frames, and $\omega_k \sim N(0, Q_k)$ and $u_k \sim N(0, R_k)$ are state model noise and measurement noise, respectively. Q_k and R_k are proportional to the person's size. Q_k is also inversely proportional to the number of tracked frames.

When processing a new video frame, we employ the method in [50] to associate each bounding box with a tracker. In the association process, we leverage the observation that position, direction of speed, and size, are not likely to change significantly in adjacent frames. In fact, the higher the speed of the user,

the larger the chance that the direction of speed remains the same. Thus, after association, each target's Kalman filter goes one iteration further. Unlike [50], where a single particle filter is used to track the center of the target, we enrich each tracker with four corners of the bounding box. Each corner is also passed through a Kalman filter as described above. The bounding box thus filtered is later utilized for estimating walking direction and step phase detection.

Figures 3.7(a)-(e) show the results from pedestrian detection. Although reasonable, it fails to detect a person in Figure 3.7(b), falsely detects one in Figure 3.7(c), and suffers from occlusion in Figure 3.7(d). However, Figures 3.7(f)-(j) show the efficacy of applying tracking. Also, Figures 3.7(f) and 3.7(j) show that people can be tracked properly even after temporal encounters.

IsWalking and Walking Direction To detect whether the target person is walking, we use the speed of the bounding box. Since speed is one of the states in the Kalman filter, we obtain it from the tracking process described earlier. Figure 3.8 shows 2D speed of the center of the bounding box – higher speeds are denoted with a longer arrow. People in Figures 3.8(a)-(e) are walking and the person in Figure 3.8(f) is standing in place. If a person's movement has significant speed, then we can detect walking by the mean of 2D speed, s_{xy}, normalized by that person's height, during a predefined period as given by Equation 3.3.

$$s_{xy} = \frac{1}{L_P} \sum_{i \in P} \frac{\sqrt{(c_x^i)^2 + (c_y^i)^2}}{h^i} \tag{3.3}$$

where P is a collection of frames in the period; L_P is the cardinality of P; c_x^i and c_y^i are the horizontal and vertical components of the target's center speed at frame i; and h^i is the target's height at frame i. However, if the person is walking mainly along the line perpendicular to the camera's plane (Figures 3.8(d) and (e)), s_{xy} can be as small as the case where the person is just standing. To cope with this scenario, we use a term s_z to estimate the target's motion along the Z-axis.

$$s_z = \alpha_z \frac{h^{L_P} - h^1}{mean\{h^i : i \in P\}} \tag{3.4}$$

where α_z is a calibration factor such that $|s_z|$ and s_{xy} are roughly similar if the target's speed is the same whether moving in 2D plane or along the Z-axis.

When the combined value of $s_{xy} + |s_z|$ of a target is above a certain threshold, we mark that person as walking.

We calculate the user's walking direction with respect to the camera's facing direction. The relative direction is quantized to eight bins with centers at 0°, 45°, 90°,..., 315°. To classify the direction, we train a model with bagged decision tree with features s_x (Equation 3.5), s_z, and d_z, where d_z is the slope of linear regression of heights. Intuitively, the bounding box's speed and increase/decrease of its height together help in determining the user's relative direction.

$$s_x = \frac{1}{L_P} \sum_{i \in P} \frac{c_x^i}{h^i} \tag{3.5}$$

Step Duration and Phase To detect the duration and phase of walking steps, we choose the lower region of the bounding box obtained from the tracker. We represent the motion in this region through Space-Time Interest Points, which essentially captures fast changes in video pixels. This technique is borrowed from [51] and we omit the details in the interest of space. As an abstraction, the technique marks the fast-changing spots on the video (Figures 3.9(a)-(c)) with brighter spots indicating faster movements. Clearly, the distribution of spots and their brightness vary while the user is walking. We define a feature, F_{center}, which captures the rhythm of this alteration. Figure 3.9(d) shows a typical path of F_{center} – the peaks are essentially the step phase, while the time separation between the peaks is the step duration. The technique becomes unreliable when the user is walking toward or away from the camera, in which case, we refrain from extracting step duration and phase.

Rotation The technique of Space-Time Interest Points can be applied for detecting rotation as well. The key observation is that rotation causes fast changing spots to be scattered over the entire bounding box, while other activities cause the spots to be confined to a relatively smaller region. Figure 3.10 shows an example where the spots are confined to the hands when the user is moving them; in contrast, rotation causes the spots to cover the entire body. Formally, we define $X = \{x_i\}$ and $Y = \{y_i\}, (i = 1,2,..,N)$ as the X and Y coordinates of the centroid of the spots. We define the following four features – $f_1 = max\{X\} - min\{X\}$, $f_2 = max\{Y\} - min\{Y\}$, $f_3 = var\{X\}$, and $f_4 = var\{Y\}$ – and

train a bagged decision tree to classify the spot distribution. The output is a binary answer: rotation or not.

Unknown *InSight* extracts a motion alphabet from each time unit (one second). However, confusion arises when the user transitions from one action to another within that second (e.g., stationary user starts walking). We conservatively deem these time units as an "unknown". The motion string thus contains motion alphabets and unknowns interspersed with each other.

3.1.4.3 Matching Motion Strings

We now describe how motion strings obtained from sensors are matched with motion strings extracted from videos. First, since the walking direction estimated from sensors is with respect to global north, we map it to one of the eight quantized directions relative to our camera-facing direction. Then, *InSight* computes the "distance" between the two strings, similar in spirit to *edit distance*.

Specifically, denote the motion strings based on the video and sensor data as V and M, respectively, and their length L (note that the string lengths are the same). For each position i in the strings, we compare $V(i)$ and $M(i)$. The difference between them is 0 only if: (1) $V(i)$ and $M(i)$ are identical or (2) $V(i)$ and $M(i)$ both correspond to walking in identical or adjacent directions, their step durations are within a threshold ratio, and their step phase marker of $M(i)$ falls into a range calculated from step phase makers of $V(i)$. Otherwise, their difference is recorded as 1. Let D be the total difference across the length of the strings. Then we define string similarity as $(1 - \frac{D}{L})$. When comparing a video string of a person X against multiple people's sensor strings, we pick the one (say Bob's) with the single highest similarity (if any). Only if this highest similarity is above a certain threshold, then X is identified as Bob, otherwise we declare the recognition as "unsure".

3.1.4.4 Extracting Color Fingerprint

Once Bob is identified based on his motion, the server extracts and adds his color fingerprint to its repository. Thereafter, the server may be able to recog-

nize Bob from an image without seeking video and sensor data, saving both latency and bandwidth. While various visual features of Bob can be considered to form his color fingerprint, clothing is an obvious choice as a temporary fingerprint. Therefore, in this section, we extract the features of Bob's clothing and use them as Bob's color fingerprint. As a first step in getting the fingerprint, we detect the clothing area in the person's image. Then, we use a well-known technique, namely spatiograms, to extract a color fingerprint.

Clothing Area Detection We extract color fingerprints when target is in near-front or near-back view. The Calvin upper-body detector [52] model is trained to return bounding-boxes fitting the head and upper half of the torso. Figure 3.11(a) and Figure 3.11(b) show the detected upper-body; Figure 3.11(c) shows that the detector doesn't fire in other views. We then use a pose estimation model [53] trained on Buffy dataset [54], which returns with joints such as neck, shoulder, etc. The neck and shoulder joints (red lines in Figure 3.11) help crop the upper-torso area as the clothing area. Then, we apply spatiograms on the cropped image to extract the target's color fingerprint.

Spatiograms Spatiograms are essentially color histograms with spatial distributions encoded in its structure. Put differently, while basic color histograms only capture the relative frequency of each color, spatiograms capture how these colors are distributed in 2D space. The second order of spatiogram can be represented as [55]:

$$h_I(b) = < n_b, \mu_b, \sigma_b >, \quad b = 1,2,3,\cdots,B$$

where B is the number of color bins, n_b is the number of pixels whose value falls in the b^{th} bin, and μ_b and σ_b are the mean vector and covariance matrices of the coordinates of those pixels, respectively. Through such a representation, a white over red stripe can be distinguished from a red over white stripe, even if the number of red and white pixels are identical in both. Also, to cope with various viewing distances, we normalize the spatial information with respect to the *shoulder width* so that all the spatial representation is relative to the captured body size in each photo. Finally, to decouple lighting conditions from the colors, we convert the pixels from RGB to HSV, and quantize them into $B = 5 \times 1 \times 2$ bins.

3.1.4.5 Color Fingerprint Matching

When John views Bob through his glass – either from the front or the back – *InSight* again crops out a region around Bob's upper body, and applies the same fingerprinting operations on this image. These fingerprints – one in the repository and another from John – are now ready for matching. Our matching algorithm first computes the spatiogram similarity between each person in John's view with Bob's fingerprint in the repository. Denote the spatiograms to be compared as $S = \{n, \mu, \sigma\}$ and $S^{'} = \{n^{'}, \mu^{'}, \sigma^{'}\}$, both having B color bins. We define the similarity measure as in [56]:

$$\rho = \sum_{b=1}^{B} \sqrt{n_b n_b^{'}}\, 8\pi |\Sigma_b \Sigma_b^{'}|^{1/4}\, \mathcal{N}(\mu_b; \mu_b^{'}, 2(\Sigma_b + \Sigma_b^{'}))$$

Essentially, the similarity decreases (following a Gaussian function) with increasing difference between the colors and their spatial locations. Fingerprints are considered to match if ρ is greater than a certain threshold.

When motion information (video and sensor data) is available in addition to clothing fingerprints, *InSight* server utilizes them both to make the recognition more robust. First, it computes the ρ value for each video frame that captures target's near-front or near-back view, and calculates the mean $\bar{\rho}$. It deems clothing similarity as 1, if $\bar{\rho}$ is above a certain threshold and 0 otherwise. Next, it will compute the overall similarity as the average of motion similarity and color similarity. Then, it will pick a person with the single highest overall similarity. If this person's overall similarity is above a certain threshold, then *InSight* returns the person's name, and unsure otherwise.

3.1.5 Evaluation

This subsection is organized in three parts: **(1) Micro-benchmarks** to evaluate the accuracy with which motion alphabets can be detected from each second of video and sensor data. **(2) Scenario with real users** to evaluate *InSight*'s ability to discriminate individuals through motion/visual fingerprint comparison. **(3) Video simulation** to evaluate scalability across large number of users. The experiment design and details are presented under each of the three parts.

3.1.5.1 Micro-Benchmark (for Motion Alphabets)

Experiment Design Motion alphabets define the atomic operation in *InSight*. This subsection evaluates whether each second in videos and sensor data can be reliably converted to motion alphabets. For this, we recruited 12 volunteers, gave each of them a Samsung Android phone running an *InSight* client, and asked each of them to cover various actions, such as walking, rotations, taking turns, standing still, etc., as well as some upper-body movements like checking emails, stretching, etc. in an area of 20 m × 15 m outside our building. During the entire experiment, a designated observer video-recorded each volunteer's motion patterns separately. The experiment was performed in four sessions and in total 80 minutes of sensor and video data were collected (and each frame manually labeled for ground truth). These videos were then examined frame by frame and manually labeled with ground truth (i.e., each second of the video was tagged with one of the motion alphabets, such as walking or not, walking direction, starting and ending frame of each step, rotating or not, etc.).

Results

Walking Detection We use a three-fold cross-validation to evaluate the accuracy of walk-detection. Recall that walking detection with sensors is based on bagged decision trees, while for videos, we used a calibration factor α_z and a motion threshold. We set $\alpha_z = 4$ and motion threshold of 0.5. Figure 3.12 shows the confusion matrix – evidently, the detector is highly accurate for both sensors and videos, with mis-detection not above 0.6% and 1.5%, respectively.

Step Duration We painstakingly computed the ground truth for step duration, i.e., the start and end time of each step. This enables comparison with estimates made from video and sensor data. Figure 3.13 plots the CDF of the relative error for both dimensions of information. The error distribution with videos exhibits a staircase function since the ground truth was marked in the units of video frames. Overall, the relative error is less than 8% in more than 85% instances for both video and sensor data.

Walking Direction Using sensor data, we compute the walking direction w.r.t. global north and plot the relative error in Figure 3.14. Evidently, the error is not high and confined mostly to $\pm 45^\circ$ around the true value. We also estimate

the walking direction from the videos and classify them into one of eight classes – Figure 3.15 reports the confusion matrix. Classification accuracy (at 45° granularity) is consistently high, and the slight confusion is mostly with adjacent angular directions.

Step Phase Recall that step phase of two individuals is the difference between the time points at which their respective feet strike the ground. Figure 3.16(a) shows the histogram of the difference in the phase (normalized by T_{step}), estimated from the sensor data and compared against ground truth. Figure 3.16(b) shows the same histogram but computed from video data. However, for recognizing individuals in *InSight*, what matters is the video-sensor phase offset (i.e., the difference in phase computed between video and sensor data). Figure 3.16(c) shows this difference. The graph suggests that the resolution at which step phases can be discriminated is around $0.3 \times T_{step}$ (T_{step} is the step duration) – two individuals that are different by less than this value will appear to be in lock-step.

Rotation Recall that, rotation is detected from videos by training a bagged decision tree and classifying the identified spot distribution (Figure 3.10). For extracting rotation from sensor data, we compute the rotation angle around gravity, $\alpha_{gravity}$, and apply a threshold of 15°. Figure 3.17 reports the a confusion matrix. The high accuracy confirms reliable rotation detection.

3.1.5.2 Real User Scenario

The above evaluation shows the consistent accuracy of detecting motion alphabets from both video and sensor data. We extract motion strings from individuals and examines the discriminative abilities in them below. The overall performance depends on two factors: (1) the inherent diversity in human motion patterns, and (2) effectiveness of *InSight*'s fingerprint design and matching schemes.

Experiment Design We again conduct experiments with the help of the same 12 volunteers as before. But, unlike the micro-benchmark setting, in this set of experiments, the volunteers' motion and behavior were completely natural. They were allowed to naturally move around or pause, and do as they pleased.

They were also not instructed about clothing; they came to the experiment wearing the same clothes that they wore to school that morning.

We have not developed a real-time version of *InSight*. Our current evaluation is offline and structured as follows. We pretend that the observer requests to recognize one of the volunteers at a random time t. Starting at time t in our dataset, we crop out a 10 second video of that volunteer, as well as a 10 second sensor stream from all 12 smartphones. A motion string derived from this video is then matched against all the motion strings derived from the sensors. The string matching algorithm either returns a matching smartphone (or volunteer), or returns *unsure* if there is no single highest score or the score is below a threshold (set to 0.95). We repeat this experiment 100 times with different request times.

Results Figure 3.18 shows the recognition accuracy. To understand the contribution of different motion alphabets toward overall human recognition, we evaluate various combinations of alphabets. Figure 3.18(a) starts with the simplest one – walking or not. In other words, for each second of the video and sensor data, a volunteer's motion is categorized as walking or not walking. Then, for increasing string lengths, we plot the performance of the matching scheme. For increasing time durations (on the X-axis), we show the fraction of people correctly recognized, incorrectly recognized, and unsure. For instance, from the first seven seconds of video and sensor data, we could correctly recognize 2% of the cases and the rest were unsure (none incorrectly recognized). As we consider longer durations of motion, recognition performance improves, reaching 7% with 10 seconds of motion. This improvement is expected since motion of two individuals diverges over time making them more distinguishable. Of course, the distinguishability is not high in this case, since *walking* alone, is hardly a strong discriminator.

To improve over a binary walking indicator, we include *walking direction* in the motion alphabet (quantized to eight classes) and present its performance in Figure 3.18(b). Understandably, walking direction helps improve the recognition, up to 30%. Similarly, Figure 3.18(c) and Figure 3.18(d) indicate that considering step phase and duration together along with walking direction can recognize individuals correctly in 50% of the cases. Figure 3.18(e) shows that when rotating or not is further added to the alphabet (in addition to walking direc-

tion/phase/duration), a person can be recognized in close to 72% cases with 10 seconds of observations.

Finally, clothing fingerprints can also be used in combination with motion based alphabets for recognizing humans. Figure 3.18(f) shows that when clothing fingerprint is used in conjunction with motion, six seconds of observation are sufficient to recognize a person with 90% probability. Overall, these results demonstrate that motion and clothing together help recognize individuals accurately, and recognition performance improves over time.

3.1.5.3 Video Simulation (for Scale)

Experiment Design Recruiting a large number of volunteers and bringing them for multiple social experiments proved more difficult than we imagined. Still, to gain insights on *InSight*'s scalability to higher user density and different settings, we resort to an approximation. Our key idea is to record video of people in public places, and even though we do not have sensor data from them, we will *synthesize sensor data by injecting statistical error into ground truth observations*. For this we execute the following steps:

1. Record videos of people in public places, such as university cafes, grocery stores, busy street intersections.

2. Extract the video-based motion fingerprints for each user in the video, denoted V_i.

3. Manually extract ground truth for each user from the video, denoted T_i.

4. Inject errors into the ground truth based on past error distributions, observed when extracting motion alphabets from sensors (i.e., $M_i = T_i + Error$).

5. Compare the video-based motion strings to the synthetic sensor based strings (i.e., $V_x == M_i$?).

The results should be a faithful approximation of *InSight*'s performance at scale.

The motion alphabets are synthesized as follows. To determine whether a person is walking or not, we use the confusion matrix shown in Figure 3.12(a).

Specifically, if a person is walking as per the ground truth, then the person is marked as walking with 99.4% probability and as not-walking with 0.6% probability; when the person is not walking, the motion is marked as walking and not-walking with 0.1% and 99.9% probability, respectively. We follow the same for deciding rotating or not, using the rotation confusion matrix in Figure 3.17(a). To obtain the step duration, we add a random error to the ground truth, following the distribution in Figure 3.13. For step direction, we do the same using the distribution from Figure 3.14. For determining the step phase, we add a random shift according to the step phase distribution in Figure 3.16 (a). The random shift varies from -9.1% of T_{step} to 25.3% of T_{step}. Now, since the error distributions were from a 12 member evaluation, the variance is likely to be smaller compared to a larger population. To account for such situations, we added additional errors to ensure the data is not optimistic.

The public videos were recorded under three different scenarios – near a busy area outside the student union during summer (referred to as the "union" video later), at the CS department cafe in the winter (referred to as the "cafe" video), and at the entrance of a store in the winter (referred as the "store" video). Figure 3.19 shows example video frames. People moved in and out of the videos – so at any typical time instant, we observed between 3 to 10 in the view finder. However, for each of the videos, we computed motion fingerprints of all the people across time, and then compare against each other. For instance, the union video included 40 distinct individuals in five minutes, and we pretend as if all the 40 people were present at the same time. *InSight* is expected to be able to discriminate each individual accurately from these 40 individuals. The cafe and store had 15 people each due to less churn.

Results Figure 3.20 reports results from the "union" video simulations. As a high-level summary, *InSight* was able to recognize most of the users by combining motion and clothing. Specifically, since the recording was in summer, people wore colorful clothing, which by itself achieved 50% accuracy among 40 people. Expectedly, motion aided this discrimination; however, given that many users walked often, walking-or-not was not a major discriminator. Walking direction and step duration were helpful, but still not sufficient due to the high density of users (users mostly walked in two dominant direction, one to-

ward the restaurants, and another toward dorms). However, when including walking phase, results improved significantly.

We zoom into the results here. Figure 3.20(a) shows the clothing confusion matrix across 40 people. Figure 3.20(b) and Figure 3.20(c) plot the recognition performance over time, using motion alone followed by motion+clothing. Figure 3.20(d) shows the confusion matrix after combining motion and clothing and the similarity threshold set to 0.95. Using clothing alone, *InSight* recognizes 50% of individuals. Using motion alone, *InSight* recognizes 40% of individuals within 10 seconds; with the combination of clothing and motion, the recognition performance rises up to 88% within 5 seconds and 90% within 8 seconds. Figure 3.21 and Figure 3.22 report on the results from the "cafe" and "store" video simulations. Since both scenarios were recorded in winter, a majority of the people were wearing dark shades (dominantly black and gray). Thus, in both cases, clothing offered fewer bits of information. However, motion compensated well, especially in the cafe where people walked, paused, turned, etc. With combined clothing and motion, *InSight* achieves 80% accuracy in 6 seconds, 93% in 8 seconds at the cafe, and 87% in 5 seconds, 93% in 10 seconds at the store.

The results suggest that humans inherently exhibit diversity in their motion and clothing patterns that even low-resolution feature vectors offer promise of identification. Where faces are permanent and high entropy fingerprints, these motion strings could serve as a useful alternative, when temporary "visual identification" is of interest, without revealing persistent identities.

3.1.6 Points of Discussion

We discuss a number of limitations and untapped opportunities with *InSight*.

Coping with Practical Hurdles While we believe *InSight* is amenable to deployment in the real world, it needs to be engineered, tested, and fine-tuned for various practical scenarios. Occlusions is perhaps the key limitation at this point. In a crowded environment, an individual is likely to be occluded by others, preventing the computer vision algorithm from carving out a precise bounding box for each person. This can inject confusion in the system, especially if the bounding boxes erroneously include different parts of two or more

individuals. Coping with these complications is left to future work. Also, low lighting intensity in certain environments may affect visual clarity and fingerprinting. Further, people may change clothing, such as put on a scarf or take off their jackets, after their fingerprints have been registered in the database. Of course, motion fingerprints are a dependable fallback to cope with some of these situations; however, their efficacy in the wild remains to be seen.

Real Time Operation In its current form, *InSight* is an offline system. Running this online will require substantial "heavy lifting" likely to be executed in the cloud [57] or cloudlets [58]. This section's focus is toward demonstrating the core opportunity, that is the necessary engineering for an end-to-end system will depend on the application in question, and is a separate work altogether. Innovative research is yielding intelligent cloud-offloading techniques [57, 58], designed explicitly for applications like *InSight*. In view of this, we believe real-time operation will be feasible over time, perhaps requiring porting *InSight* on MAUI-like programming frameworks [57].

Energy The energy footprint of *InSight* may not be excessive. Motion sensors on smartphones need not consume much energy even after prolonged continuous sensing. The camera on the other hand can be activated only when the user desires to view annotations of the environment (e.g., when Alice wants to learn the identity of a particular person). This is true even for other applications such as privacy preserving pictures; the camera is again used only during the recording of a video or for taking a picture.

Incremental Deployment If some users do not run *InSight* on their phones, the fingerprint matching process faces additional challenges. For instance, Alice may be looking at Bob in reality, and even though Bob is not running *InSight*, Bob's clothing fingerprint may match best with Chris (a registered user). To avoid such errors, *InSight* requires that the fingerprint matching threshold be very high, with the hope that motion-based matching would be triggered. Nonetheless, certain scenarios (such as weddings, funerals, uniformed school children) may still derail *InSight*. Perhaps an adaptive process is needed, where the *InSight* server identifies large similarities in clothing patterns and triggers motion-based matching more aggressively.

Utilizing All Bits of Information Upon receiving a video clip, *InSight* can extract clothing fingerprints for multiple people even if it is unable to map them to their identities. Over time, the repository of unnamed clothing fingerprints increases. Now, as individuals get identified through motion, it may be possible to resolve some of the other clothing fingerprints via the process of elimination. We envisage that such a form of inferencing would be possible due to the global view on the fingerprint data, available at the *InSight* server. We have not tapped this opportunity in this section.

Peer-to-Peer Version of *InSight* While we describe *InSight* as a cloud-based system, it is also possible to realize it over a peer-to-peer model. The sequence of operations, somewhat different from Figure 3.2, can be as follows. (1) Alice's glass takes a picture of X and broadcasts it, asking "who is the person in the picture?"; simultaneously it starts recoding a video snippet. (2) Smartphones in the vicinity receive the image (over Bluetooth or WiFi Direct), extract the color fingerprint, and match it with their self color fingerprint (if they have one); simultaneously each of them activate their motion sensors. (3) These smartphones send their color fingerprint matching score along with their sensor data to Alice's glass. (4) Alice's phone computes the motion matching score, combines it with the color score, and ultimately identifies X as Bob. (5) Alice's glass extracts Bob's color fingerprint from the video snippet and unicasts it to Bob's smartphone, which updates its own fingerprint. Of course, such a system assumes that smartphones are capable of executing the compute-heavy algorithms locally, which is perhaps difficult in today's platforms.

3.1.7 Related Work

There exist several works on activity recognition based on video [59, 51] and sensors [60]. TagSense [61] uses motion as an indication of whether a person is in the picture, but does not need to actually identify each individual. Face recognition and other visual bio-metrics are, of course, possible alternatives [42, 62, 63], but need the face to be visible (in addition to practical concerns on revealing a permanent identifier). *InSight*, on the other hand, temporarily fingerprints individuals, exposing only soft-biometrics of the user that cannot identify them later. We observe that *InSight* is different from gait analysis [63], used to "fingerprint" individuals. While gait analysis zooms into the intricacies

of walking patterns (i.e., how one walks), we use far higher-level motion alphabets capturing duration of walks, turns, pauses (none of which is permanent). Authors in [64] proposed the usage of walking speed extracted from the infrastructure camera and wireless accelerometer sensor node worn on users' belts. *InSight* combines a rich set of motion alphabets and clothing colors to boost the power of the "fingerprint". In addition, *InSight* fuses multimodal sensor data on off-the-shelf smartphones to bypass the needs for wearing customized sensor nodes on users' belts.

The specific applications we have discussed have received research attention in the recent past [65, 66, 67, 68, 69, 70]. Researchers have explored the possibility of looking at objects in a store, using wearable devices and radio-optical beacons [65] and/or RFID-based techniques [67]. Such modes of communication are innovative and complementary to *InSight*. Qualcomm Vuforia [71] is a commercial Mobile AR SDK for object recognition and 3D object tracking. Videoguide [72] is a Vuforia app used to animate architecture work in the Barcelona museum. Contrary to Vuforia which requires deployment in advance, *InSight* is a training-free system intended for humans.

Privacy preservation in the age of wearable cameras is also witnessing considerable research attention. Authors in [45] suggest a QR code pasted on people's clothing, as an expression of privacy preferences to surrounding cameras. Of course, the QR code may not necessarily be in view of the camera; reading from longer ranges is difficult; with human motion, reading QR codes is difficult. PrivateEye [73] proposes to avoid recording objects by drawing a signature shape around it. *InSight*, on the other hand, does not require any form of instrumentation, except that the smartphone should run the app. Natural behavior of humans should exhibit adequate diversity for recognition, in turn useful for inclusion, exclusion, or communication.

Our workshop paper on *InSight* [8] was an initial exploration into the possibility of using clothing colors as a temporary visual identifier. This section builds on the workshop version in multiple fronts, including (1) the significant addition of motion information to the notion of visual fingerprinting, (2) a range of techniques to correlate motion from vision and device sensors, (3) the idea of expressing fingerprints as activity–strings and applying string matching algorithms on them, (4) a more complete evaluation through micro-

benchmarks and offline evaluation, and finally (5) isolating the notion of visual fingerprinting and discussing its various applications in augmented reality, privacy-preserving pictures, and indoor localization.

3.1.8 Conclusion

This section pursues a hypothesis that motion patterns and clothing colors may pose as a human fingerprint, adequate to discriminate one individual from others. If successful, such a fingerprint could be effectively used toward human recognition or content announcement in the visual vicinity, and more broadly toward enabling human-centric augmented reality. Pivoted on this vision, we develop a proof of concept – *InSight* – which combines the motion fingerprint and clothing colors for human recognition. We find promise in this direction, and are committed to building a fuller, real-time system.

3.2 Localization Accuracy

To evaluate the localization accuracy of VideoLoc, we conduct experiments with the help of six volunteers. Each of them naturally moved around and did as they pleased in an area of 100 m^2 in our engineering lobby for 1.5 mins. A designated observer video-recorded volunteers (see Figure 3.23). We manually labeled eight pairs of points from a video frame and the building floorplan to find the projective transform between the two frameworks. Note that this manual labeling is a one-time effort as long as the camera does not move. Using this transform, we were able to calculate the real-world location for any given pixel in the frame. The video was then examined frame by frame and manually labeled with the location for each subject. These labeled locations were transformed to the floorplan framework and used as ground truth.

We borrow existing computer vision techniques [47, 48, 74, 75, 76, 77] to track the subjects and estimate their locations. Figure 3.23 shows a typical example of localization results. Figure 3.23(a) shows the estimated locations in the image framework. Figure 3.23(b) shows the estimated locations in the floorplan framework.

Figure 3.24 plots the CDF of localization error. The average across all six subjects is shown in black, while the localization error for each individual subject shown in gray. We observe that the median error is 0.11 m, while for 99th percentile, the error is 0.58 m.

3.3 Discussion

Using surveillance cameras to track humans is certainly not new [47, 48, 74, 75, 76, 77, 78, 79, 80, 81, 82]. Our preliminary evaluation using existing computer vision techniques [47, 48, 74, 75, 76, 77] confirms that video-based localization holds promise for high accuracy. However, the key question in VideoLoc is how to send the estimated location from the camera to the user's smartphone. To enable such communication, we need the user's visual address. While the user's face [42, 43], gait [63], walking speed [64], etc. could be possible approaches, we found they may not be always sufficient. We developed our core technique – *InSight* – which is a general "visual addressing" technique and is the main focus of this chapter. In VideoLoc scenario, we can leverage the geographical knowledge of the scene to fully unleash the power of "visual address". We leave this opportunity to future work.

3.4 Figures

Figure 3.1: Example social event: Alice views people's posts displayed in her Google Glass. The whole operation is orchestrated by the *InSight* server running in the cloud.

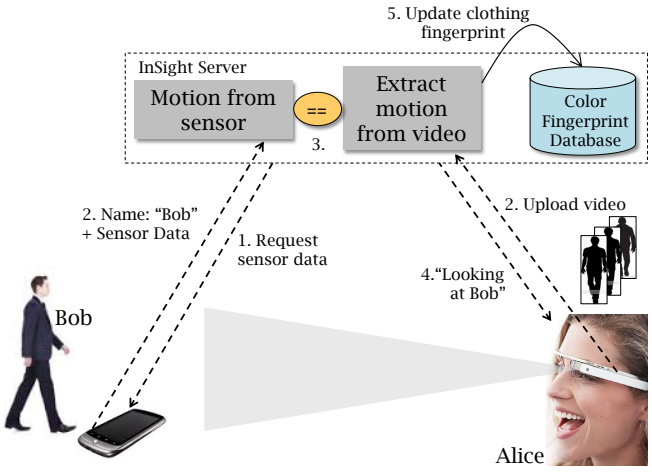


Figure 3.2: Motion fingerprint-based matching during initialization and for new *InSight* users.

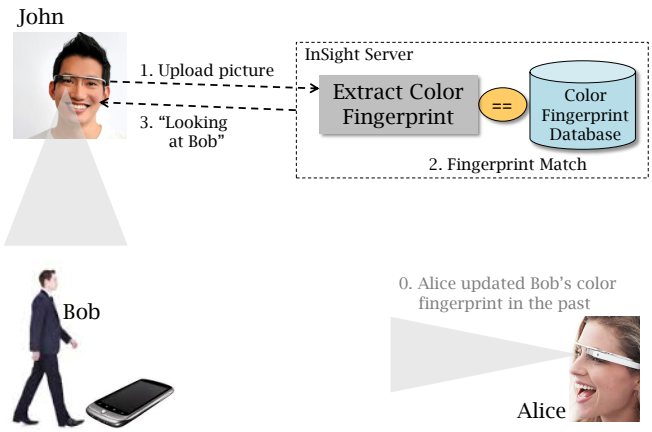


Figure 3.3: Color fingerprint matching in steady state.

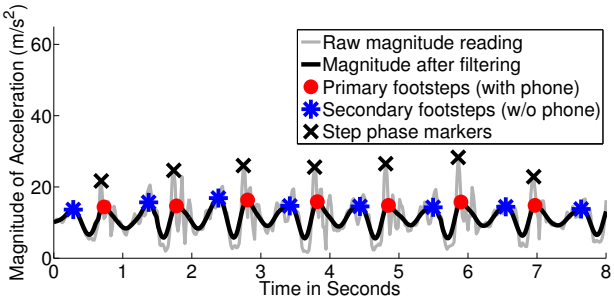


Figure 3.4: Magnitude of the accelerometer readings (smoothed) with the step marker while the user is walking.

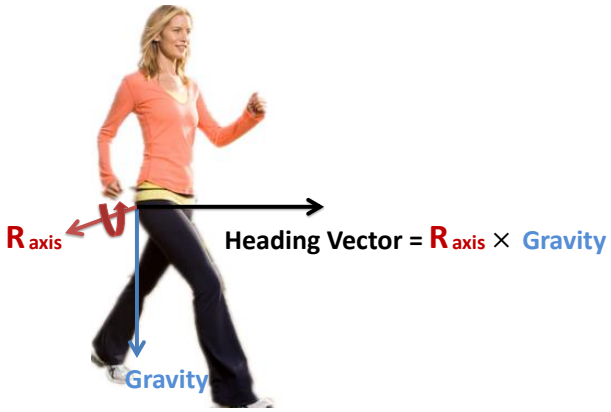


Figure 3.5: Heading vector = cross product of rotation axis and gravity.

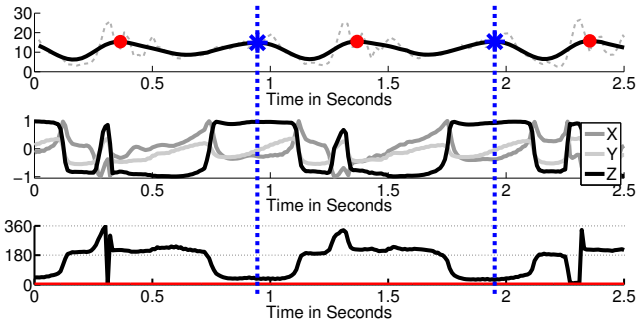


Figure 3.6: Walking direction. (top) Magnitude of acceleration while user is walking. (middle) Heading vector. Dotted lines are the moments when heading vector is calculated. (bottom) Estimated walking direction (ground truth of 0° in red). As expected, they match at the primary steps and differ by 180° at the secondary steps (with heading direction opposite of the primary steps).

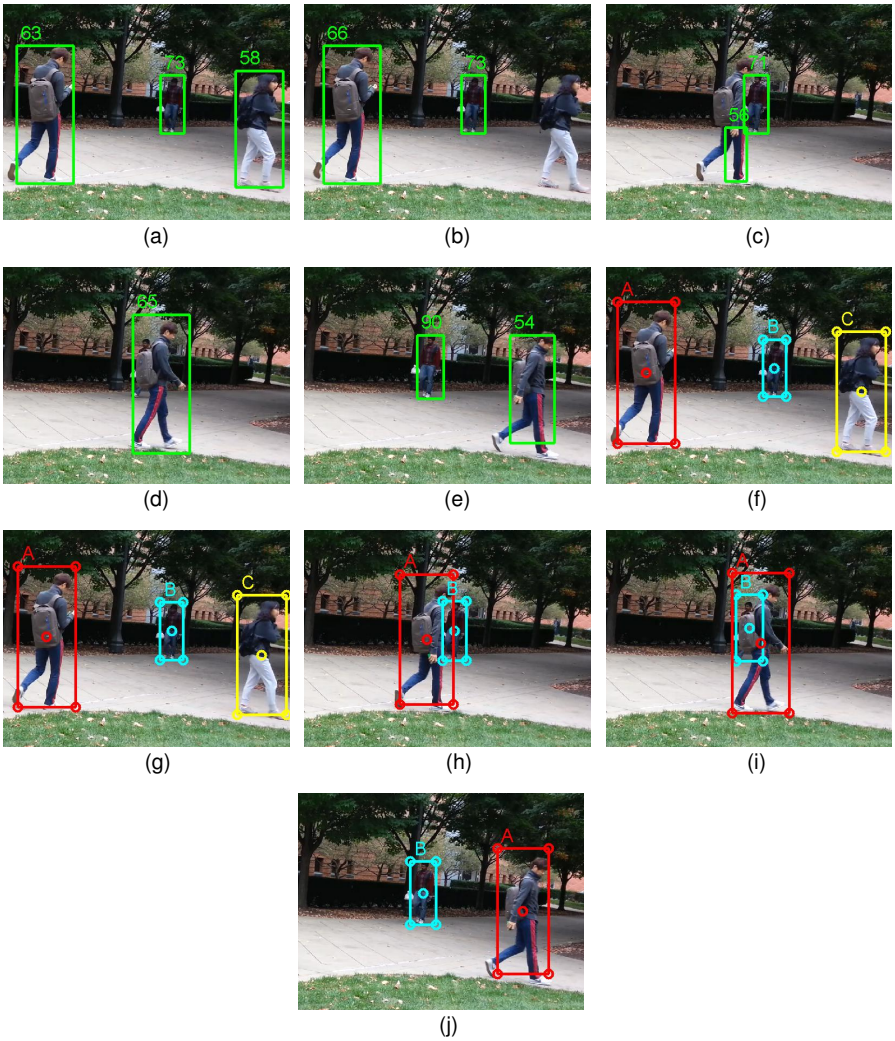


Figure 3.7: Pedestrian detection and tracking. (a)-(e) Pedestrian detection results. (f)-(j) Tracking results. (b) A temporal miss. (c) A temporal wrong detection. (d) A miss caused by occlusion.

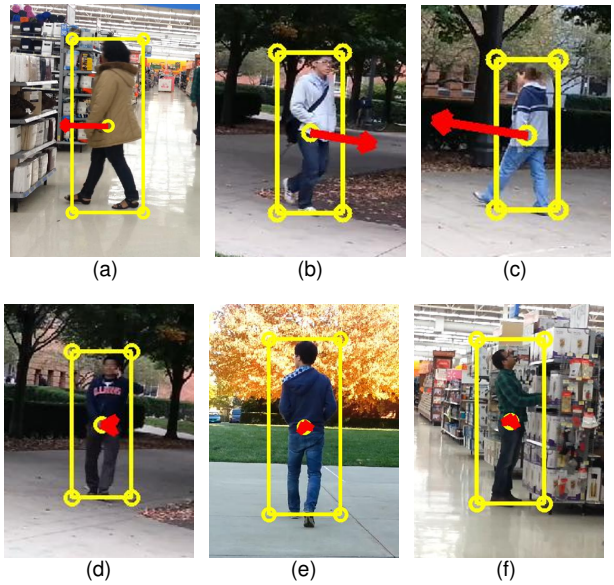


Figure 3.8: Center speed in 2D plane. (a)-(c) Cases with significant speed. (d)-(f) Cases with insignificant speed.

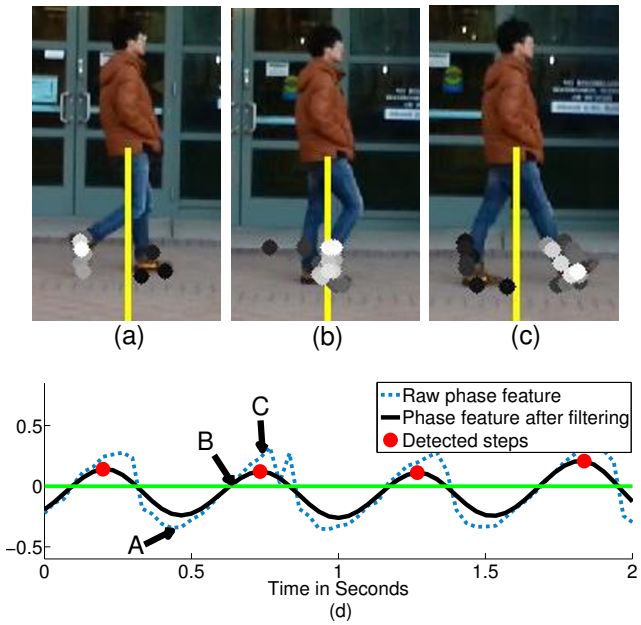


Figure 3.9: Walking phase and duration. (a)-(c) Centers of detected cuboids. (d) The walking phase feature and the detected steps. Points A, B and C in (d) correspond to the instances in (a), (b) and (c), respectively.

Figure 3.10: Scattering of motion spots. (a) Rubbing hands. (b) Putting hands into pocket. (c) Rotating.

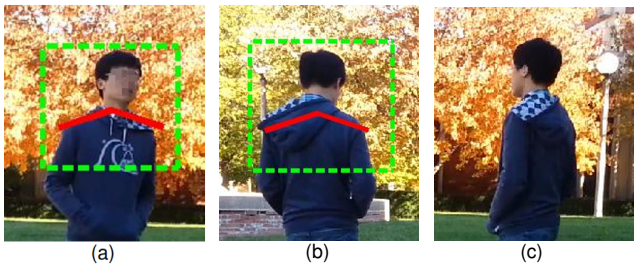


Figure 3.11: View detection and shoulder estimation. (a) Near-front view. (b) Near-back view. (c) An example of the other views. Green dashed boxes show the detected upper-body. Red solid lines are shoulder extracted by pose estimation.

	No	Yes
No	99.9%	0.1%
Yes	0.6%	99.4%

(a)

	No	Yes
No	98.9%	1.1%
Yes	1.5%	98.5%

(b)

Figure 3.12: Walking or not. (a) Results by sensors. (b) Results by vision.

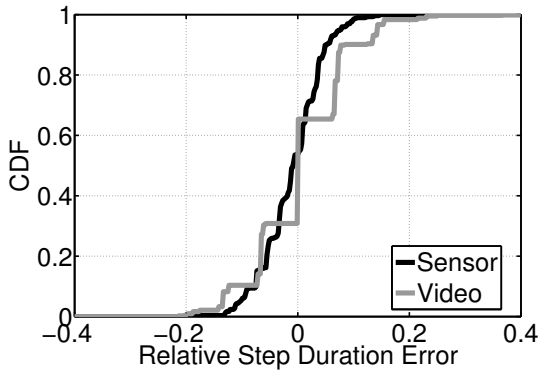


Figure 3.13: Step duration estimation error compared to ground truth.

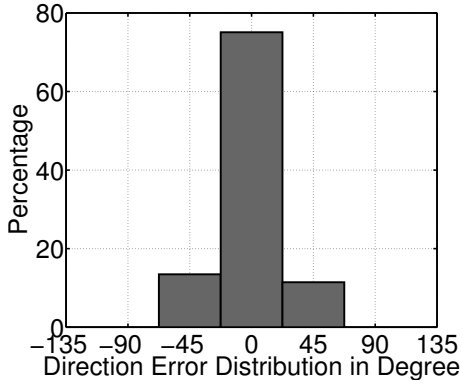


Figure 3.14: Walking direction estimation: relative error with sensors.

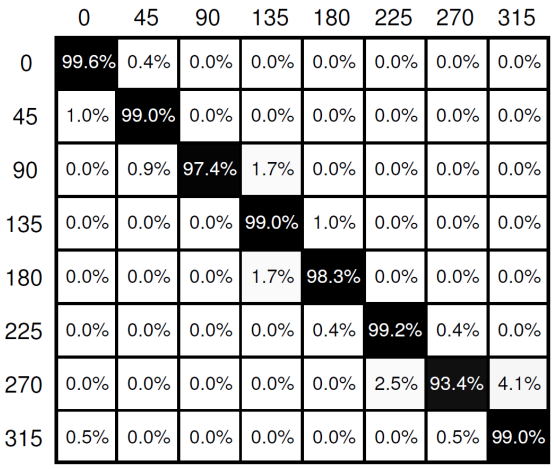


Figure 3.15: Walking direction estimation: confusion matrix for quantized directions based on video.

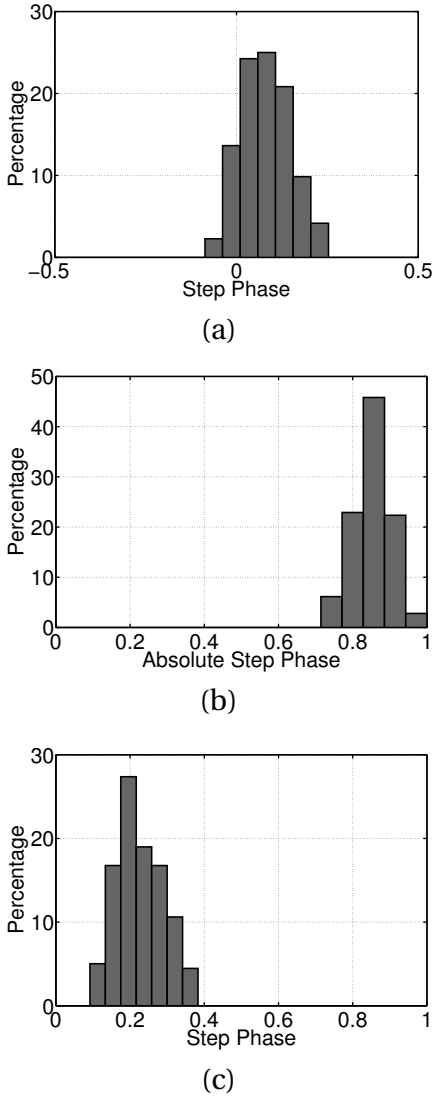


Figure 3.16: Step phase marker distribution. (a) Sensor phase markers with respect to ground truth. (b) Video phase markers with respect to ground truth. (c) Sensor phase markers with respect to video phase markers.

	No	Yes
No	97.8%	2.2%
Yes	1.3%	98.7%

(a)

	No	Yes
No	98.9%	1.1%
Yes	1.2%	98.8%

(b)

Figure 3.17: Rotating or not. (a) Results by sensors. (b) Results by vision.

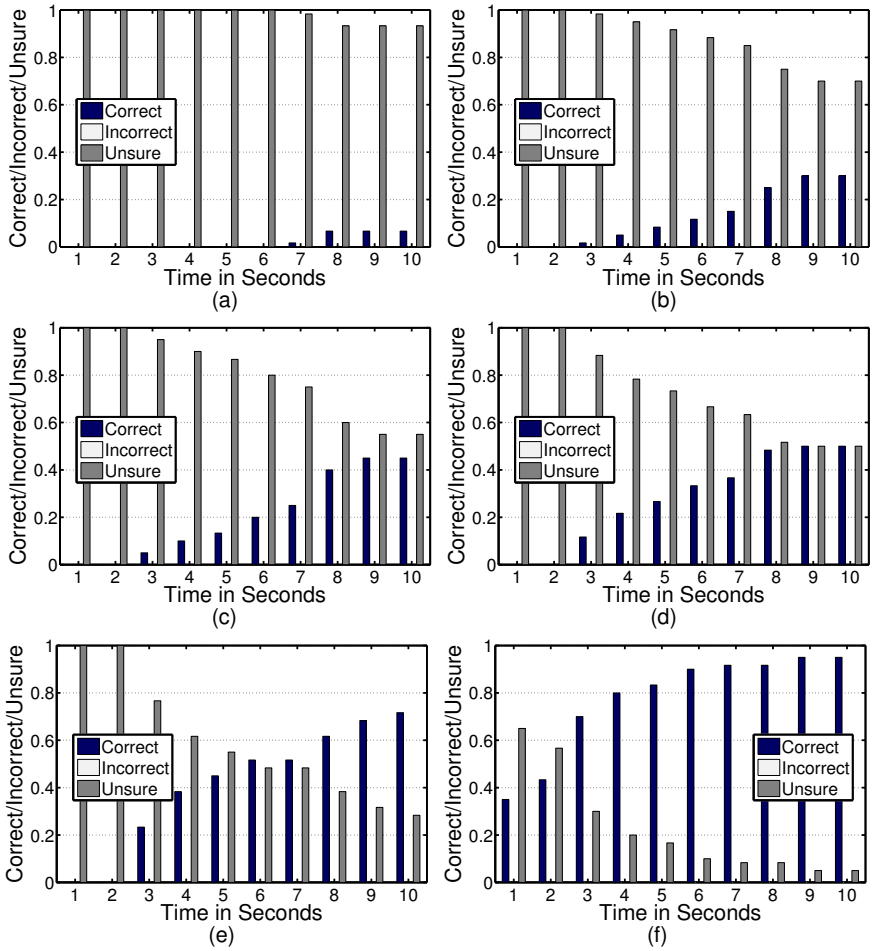
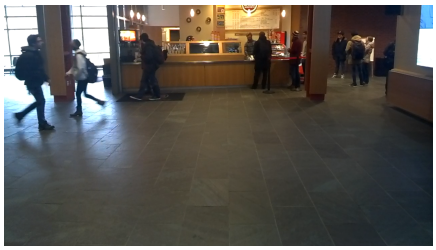


Figure 3.18: Recognition performance with motion alphabets and color fingerprints. (a) Walking or not only. (b) Walking direction only. (c) Walking direction with walking phase. (d) Walking direction with step phase and duration. (e) All motion alphabets including rotation. (f) All motion alphabets along with color fingerprints.

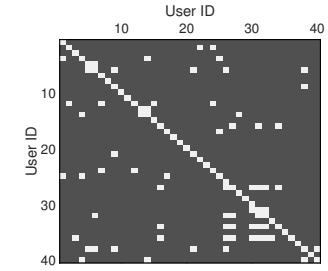
(a)



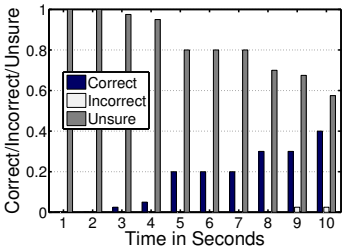
(b)

(c)

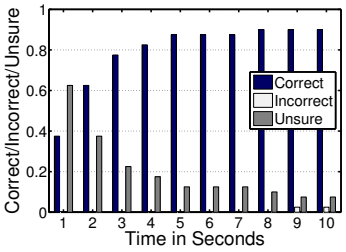
Figure 3.19: Example frames. (a) Outside student union. (b) University cafe. (c) Store.



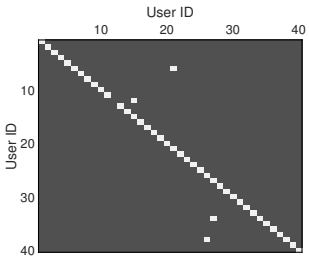
(a)



(b)



(c)



(d)

Figure 3.20: Union video performance. (a) Clothing only confusion matrix. (b) Motion only. (c) Motion + clothing. (d) Motion + clothing confusion matrix (after applying a threshold of 0.95 on similarity score).

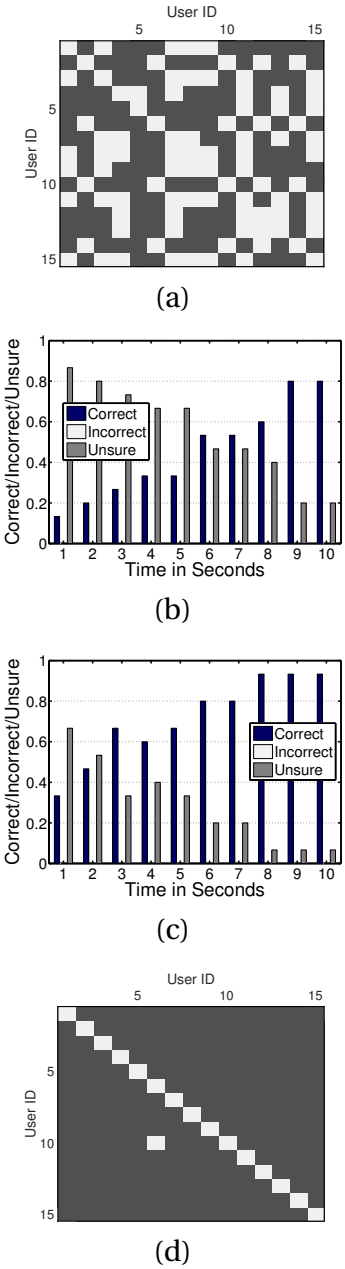


Figure 3.21: Cafe video simulation. (a) Clothing only confusion matrix. (b) Motion only. (c) Motion + clothing. (d) Motion + clothing confusion matrix (after applying a threshold of 0.95 on similarity score).

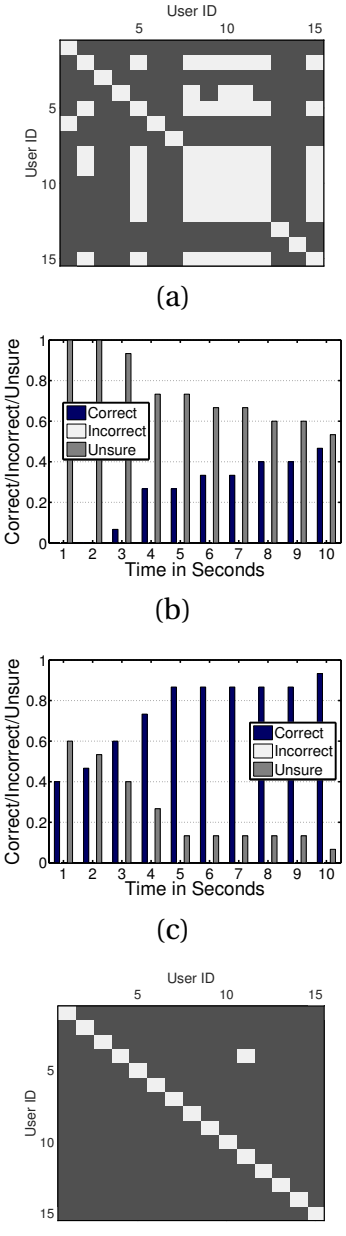
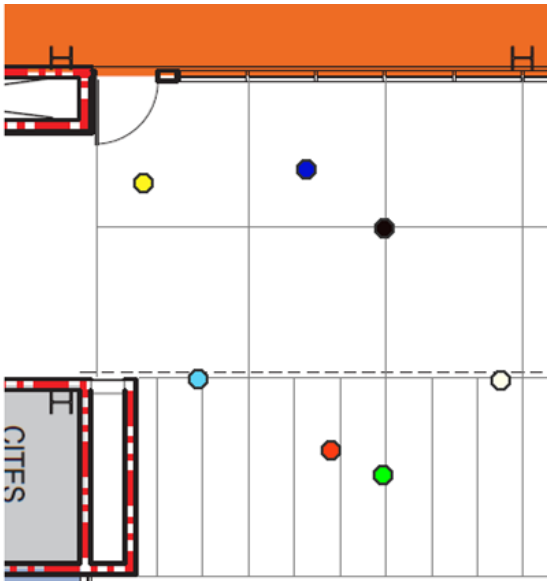


Figure 3.22: Store video simulation. (a) Clothing only confusion matrix. (b) Motion only. (c) Motion + clothing. (d) Motion + clothing confusion matrix (after applying a threshold of 0.95 on similarity score).

(a)



(b)

Figure 3.23: A typical example of localization results. (a) Bounding boxes with labeled numbers are our subjects. Red dots show the estimated locations. (b) Estimated locations transformed to the floorplan framework.

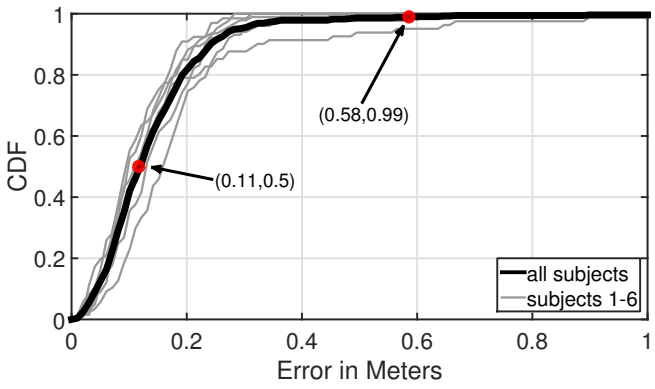


Figure 3.24: CDF of localization error for each of the six subjects.

3.5 Tables

Table 3.1: A few examples of motion alphabet. *InSight* uses much more fine-grained alphabets.

Motion Alphabet	**Explanation**
O	stationary, paused
N	walking north
S	walking south
E	walking east
W	waling west
R	rotating
U	undetermined motion

CHAPTER 4

MOLE: MOTION LEAKS THROUGH SMARTWATCH SENSORS

Imagine a user typing on a laptop keyboard while wearing a smartwatch. This chapter asks whether motion sensors from the watch can leak information about what the user is typing. While it is not surprising that some information will be leaked, the question is *how much?* We find that when motion signal processing is combined with patterns in English language, the leakage is substantial. Reported results show that when a user types a word W, it is possible to shortlist a median of 24 words, such that W is in this shortlist. When the word is longer than six characters, the median shortlist drops to 10. Of course, such leaks happen without requiring any training from the user, and also under the (obvious) condition that the watch is only on the left hand. We believe this is surprising and merits awareness, especially in light of various continuous sensing apps that are emerging in the app market. Moreover, we discover additional "leaks" that can further reduce the shortlist – we leave these exploitations to future work. This chapter is published in MobiCom 2015 [10].

4.1 Introduction

Rich sensors on wearable devices are offering valuable data, enabling important applications in mobile health, user-interfaces, context-awareness, activity tracking, gaming, etc. Of course, such data is often a "double-edged sword" since it leaks information about aspects of lives that are considered private. In our struggle to define what level of data exposure is appropriate, the core question often comes to: *what can be inferred from a given sensor data?* Every so often, we find that highly surprising inferences can be made from an apparently harmless piece of data, forcing us to push back on information exposure. While this is a broad area of research, and immense work has been performed in this direction, new platforms and applications warrant a continuous vigil

on information leakage. This chapter looks into a narrow piece of this general problem. We ask: *can accelerometer and gyroscope data from smartwatches be mined to infer the words that a user is typing?* In other words, given that a user's wrist moves in the granularity of few centimeters while typing, can the corresponding motion data be used to derive the keys that the user has typed? If so, the ramifications are serious – a smartwatch app can be disguised as an activity tracker to heavily leak a user's emails, search queries, and other keyboard-typed documents. Unlike keystroke loggers that need to find loopholes in the operating system, the activity tracker malware can obtain the user's permission and easily launch a side channel attack.

This is not the first work that combines motion data processing with language structure to infer higher-level semantics. Recent research has explored various systems and applications, including writing in the air [83], remote control [84], gesture-based signing and authentication [85, 86], smoking gestures [87], etc. While all these systems bear similarity in abstraction, unique challenges (and opportunities) emerge when a particular application is addressed end to end. In our case, we find that the absence of data from the right hand is a unique constraint, and so is the issue of inferring which finger executed the key-press. For a given position of the smartwatch, any one of three or four different keys could have been pressed, which could be further interspersed by an unknown number of keys pressed by the right hand. Moreover, not all users type with equal dexterity – some use their little finger far less efficiently while others use specific fingers when it comes to digits or corner keys. Finally, detecting the typed key is also a function of where the finger was previously, injecting a notion of dependency between consecutive inferences. With these and more application-specific issues, global typing or motion models do not apply. While fundamentally new signal processing or learning algorithms may not be needed, modifying existing techniques and systematically integrating them into a whole is the crux of our contribution.

Importantly, the application of typing also offers a number of opportunities that should be leveraged to improve the inference capability of the watch. For instance, the watch motion is mostly confined to the 2D keyboard plane, in contrast to 3D gestures in the air in other applications [85, 88]. The orientation of the watch is relatively uniform across various users and, in many users, moves back to a reference position while typing (the "F" and "J" keys). Finally,

knowing spelling priors from the English dictionary further helps in developing Bayesian decisions. The combination of these challenges and opportunities motivates the research, with the aim of quantifying the degree of information leakage.

This chapter develops *Motion Leaks* (*MoLe*), a completely functional system on Samsung Gear Live smartwatches. Briefly, two of the authors pretend to be attackers and type 500 words each wearing the smartwatch on the left wrist. The accelerometer and gyroscope data is used as training data, and processed through a sequence of steps, including key-press detection, hand-motion tracking, character point cloud computation, and Bayesian modeling and inference. Then, eight different volunteers are recruited, and each is asked to type 300 different English words from a dictionary. The smartwatch sensor data from the volunteers is transferred to our server, which then short-lists K words, ranked in the decreasing order of probability (i.e., the first ranked word is considered the most probable guess). The actual words typed by volunteers are then revealed and each word's rank is computed from the short-list.

We plot the distribution of rank across all the typed words. With this being the core of the evaluation methodology, we obviously test for various parameters and conditions, including different word lengths, sensor sampling rate, different keyboards, etc. Current limitations of this work include: (1) inability to infer non-valid English words, such as passwords; (2) scalability across different watch models; and (3) inability to parse sentences due to difficulties in detecting the "space bar". We have also not tested with other wearable devices, such as *Fitbits* – we believe with some customization, the attacks can be launched on those platforms as well.

The main contributions in the chapter may be summarized as:

- *Identifying the possibility of leakage when users type while wearing a smartwatch.* We develop the required building blocks through techniques in key-press detection, hand-motion tracking, cross-user data matching, and Bayesian inference.

- *Developing the system on Samsung Gear Live smartwatches and experimenting with real users.* We perform experiments across eight users and revealing how typed words can be inferred with reasonable accuracy. In-

dividuals who came to know about our results expressed a sense of alarm and suggested that the findings be disseminated publicly.

The rest of the chapter expands on these contributions, beginning with some groundwork and measurements, followed by system overview, assumptions, design detail, and evaluation.

4.2 Smartwatch Data: A First Look

To understand the problem landscape, we take a first look into the data from smartwatches. Basic questions pertain to the amount of wrist displacement for typed keys, whether displacements for nearby keys are even visually discernible, whether the displacements are consistent over time, etc. To this end, two of the authors wore a smartwatch and recorded the accelerometer and gyroscope data as they typed each character one by one. The positive X-axis of the watch is parallel to the arm and pointed toward the fingers, the positive Y-axis is perpendicular and upward, and the positive Z-axis pointed upward from the plane of the arm. To capture ground truth, we placed a phone camera right on top of the keyboard and recorded video at 30 fps (Figure 4.1). A green and a yellow sticker placed on the watch helps with tracking the watch movement by using computer vision techniques. The watch, the phone camera, and the keyboard logger were all time-synchronized via the network time protocol (NTP). The synchronization offers precise correspondence between the sensor and visual data, extending semantic meaning to the motion signals.

Figure 4.2 shows an example sequence of video frames capturing the process of typing the character "T". The left hand starts from a home position (i.e., the key "F"), moves along the +X direction to press "T", hits the key, and returns back to the home position. The yellow arrow on the arm shows the displacement of the green marker on the watch.

Figure 4.3 plots motion data from 20 different characters located on the left side of the keyboard. For each graph, the X-axis is time and the Y-axis is the displacement of the watch computed from the accelerometer's X-axis data (the accelerometer's Y-axis and Z-axis are not shown). The light gray vertical bar in each graph marks the time of the key-press, obtained from the keyboard logger. Observe that the displacements align well with the keyboard's layout. The

first row (12345) generates the largest positive displacement and the last row (zxcvb) produces negative displacement. The left fingers are initially placed on the third row (asdf), so nearly no displacement is detected for these characters. Although preliminary, these signals offer first indication of information leakage through watches.

Figure 4.4 shows the watch displacement for the same 20 keys, but in 2D space (i.e., using the combined X-axis and Y-axis data from the accelerometer). Each color represents one row on the keyboard. While some keys (e.g., 1, t, r, 4, 5) are quite isolated, others overlap strongly – in particular, "asdf", "zxcv" and "q23" exhibit the strongest overlaps. This is not surprising. Cluster "asdf" is an outcome of the fingers being on these keys in the home position – the wrist hardly needs to move when typing these keys. Similarly, the fingers move uniformly downward for "zxcv" resulting in similarity between the keys. Finally, the hand movement for "q" is similar to "2" and "3" even though they are all not on the same row. This is because the little finger is shorter, and to type the character "q", it must move as much as the ring finger must move to type "2".

Decoding characters gets more complicated when the user types a word rather than just a single character. Figure 4.5 shows the sequence of hand displacements where the word "teacher" is typed. Obvious issues emerge: The wrist motion for each character is no longer aligned with the earlier observations since the motion is relative to the previous position of the key. Observe that "e", "a" and "c" are all far away from their respective clusters detected earlier in Figure 4.4. Moreover, we did not record "h" (pressed by the right hand), rather a small random motion of the left hand during this time. Finally, real-world environments do not have cameras, and hence the data is completely unlabeled – wrong decisions about any of the keys can derail all subsequent decisions. In sum, while sensor data from smartwatches indeed encodes the human–typed information, decoding the data reliably in real-world conditions presents non-trivial challenges.

4.3 System Overview

This section presents a functional overview of *MoLe*; details of the technical building blocks will follow in Section 4.4. In the scenario of interest, we assume

that the attacker has successfully installed the *MoLe* app in the user's smartwatch and is receiving accelerometer and gyroscope data at the *MoLe* cloud server.

Figure 4.6 illustrates the flow of operations in the end to end *MoLe* system. At the backend server, the attacker types each character on a computer keyboard multiple times and computes a character point cloud (CPC) similar to the one in Figure 4.4. The operation is performed offline, and is stored separately for use later. The cloud can also be computed from multiple people (e.g., accomplices of the attacker) strengthening the robustness of the attack.

Now, when the raw sensor data from the user arrives, it is passed through a "keystroke detection" module, responsible for two tasks. (1) It detects the timing of each keystroke by analyzing the Z-axis of the sensor data – every time a user presses a key, the watch exhibits a discernible dip in the negative Z-axis. (2) It computes the net 2D displacement of the watch by processing the signal through multiple steps, including gravity removal, mean removal, double integration, and Kalman filtering. The output of the keystroke detection module is a set of $< location_i, time_i >$ tuples, where $location_i$ denotes the estimated location of the watch at $time_i$ when the key was pressed. When all the locations are plotted on the 2D plane, an *unlabeled point cloud (UPC)* emerges (note that the characters corresponding to each point in this cloud is not known). Figure 4.7 shows a comparison of the character point cloud developed offline by the attacker and an unlabeled point cloud computed from the attackee's data.

The UPC is forwarded to the "cloud fitting" module whose task is to assign approximate labels to the points in UPC. For this, the cloud fitting module obtains the CPC that was computed earlier, and scales and rotates the convex hull of the CPC to best fit the convex hull of the UPC. The output is a rotated and scaled CPC which serves as the *reference template* for decoding the unlabeled points in UPC.

A Bayesian Inference module (BIM) now accepts three items as input: (1) the template output from cloud fitting, (2) the unlabeled points from the UPC, and (3) a dictionary W of valid English words, w_i.[1] Briefly, for each valid word w_i, BIM computes the *a posteriori* probability that the unlabeled points form w_i.

[1] For practical purposes, W contains the 5000 most frequently used English words, available from [89].

For instance, if w_i is the word "dear", BIM computes the probability that the first unlabeled point is "d", the second unlabeled point is "e", and so on. The product of the probabilities is the final probability that the unlabeled points is the word "dear". BIM computes this probability for each word w_i, and outputs a ranked list of $< word, probability >$ tuples as a guess of the user-typed word. If it is a password, the attacker can now try out all the guesses above some probability threshold; if it is an email or a search query, the attacker could manually try to decode the text from the possible sets of words. Even though *MoLe* does not offer a single suggestion, the probability estimate associated to each guess dramatically reduces the search space for the attacker. Results in Section 4.5.2 will quantify this reduction from the attacker's point of view.

4.3.1 Assumptions

Before moving forward, we intend to enumerate a number of assumptions we make. These assumptions make *MoLe* inadequate for launching a real-life attack, however, we believe that the assumptions are not fundamental and can be relaxed with some more work.

- The evaluation is performed in a controlled environment where volunteers type one word at a time (as opposed to free-flowing sentences).
- We assume valid English words – passwords that contain interspersed digits, or non-English character-sequences, are not decodable as of now.
- We have used the same Samsung smartwatch model for both the attacker and the user – in reality the attacker can generate the CPC for different watch models and use the appropriate one based on the user's model.
- We assume the user is seasoned in typing in that he or she roughly uses the appropriate fingers; novice typists who do not abide by basic typing rules may not be subject to our proposed attacks.

Under these assumptions, the design details and evaluation of *MoLe* are presented in Section 4.4 and Section 4.5.

4.4 Design Details

We describe the main techniques executed by each module in Figure 4.6.

4.4.1 Keystroke Detector

Given sensor signals as input, this module is responsible for computing the time and location of each key-press present in the signal. The location is essentially a 2D vector with the origin as the "F" key on the keyboard. Aggregating all the key-press locations will yield the point cloud as discussed earlier.

4.4.1.1 Key-Press Timing

The intuition to detect key-presses is rooted in the hand's motion in the vertical direction. When the finger dips while typing a key, the wrist also undergoes a partial dipping motion, expected to reflect in the Z-axis of the watch. Figure 4.8 shows an example of the Z-axis motion when the user types the word "administrative". Using ground truth, we observe that the actual key-presses generally produce prominent peaks, however, false positives and false negatives occur. False positives occur mainly during transition from one key to another – the hand moves up slightly to make the movement, which manifests in Z-axis motion. False negatives typically arise due to subtle Z-axis motion for keys like "asdf" that can go undetected.

To cope with these issues, we use *bagged decision trees* to classify keystrokes. A bagged decision tree is an ensemble classifier that trains multiple decision trees by selecting different subsets of feature and training examples. The classifier improves the stability and accuracy by letting each subtree learn on the attacker's labeled data, applies the learning to the unlabeled data, and then computes the final results via voting. To obtain the labeled data, we first apply a simple threshold-based peak detection method [90] on the Z-axis acceleration, and label true/false detection on the attacker's template. We purposely set the peak detection threshold to be low so that we do not miss true keystrokes. Then we extract features within a time window around the labels and train the classifier.

The feature set includes: the width, height, prominence of the Z-axis peak; the mean, variance, max, min, skewness, and kurtosis for each of the three-axis displacement, velocity, acceleration, and gyroscope rotation; the magnitude of acceleration/gyroscope; and finally the correlation of each pair between acceleration and gyroscope vectors. When the attackee's sensor data arrives, we apply the same peak detection scheme and obtain many candidate keystrokes and their features. Then the classifier identifies the validity of the keystroke and selects the max value of Z-axis acceleration to denote the timing of the key-press. Figure 4.9 shows an example of the classification result of the word "administrative".

Figure 4.10 shows the detection rate for each key-press when using one author's template model to test on eight different volunteers recruited in Section 4.5.1 (and ground truth recorded by the keyboard logger software). Expectedly, the keys pressed by the right hand are largely undetected.

4.4.1.2 Key-Press Location Estimation

The core challenge here pertains to tracking the hand motion as it moves from one key to another, and from there, inferring the location of each key-press. Tracking over time is non-trivial since the required accuracy is high (in the granularity of key sizes); moreover, incorrect detection of one key will affect subsequent results. The hope we have is that the the left index finger periodically moves back to the home key "F", and hence, it is an opportunity to recalibrate the tracking process at the start and end of a sequence of typed characters.

As a first cut, we used the linear acceleration (offered by the native Android API) to compute displacement. We applied established double integration and mean removal techniques. Figure 4.11 compares the X displacement computed by Android API and *MoLe*, against ground truth (available from the camera). The errors proved inadequate for our purposes. Hence, we developed an improved tracking technique tailored to the *MoLe* application. We define each of the steps below.

1. **Find gravity to define an absolute coordinate system.** Before the attacker (or attackee) starts to type, his or her hand is stable. We use this opportunity to estimate the direction of gravity in the watch's coordinate system, and we can then estimate the orthogonal plane, which is the ab-

solute horizontal plane. We then project the watch's X-axis to the absolute horizontal plane to get the absolute X-axis. Since the absolute Z-axis is essentially along the direction of gravity, the cross product of the Z-axis and X-axis yields the absolute Y-axis. Here, $C = (X, Y, Z)$ is the absolute coordinate system (represented by the watch's coordinate system). Of course, since the wrist orientation changes during typing, this representation changes as well. Thus, at the starting point, we represent the absolute coordinates as $C(0) = (X(0), Y(0), Z(0))$.

2. **Estimate and remove gravity.** From the gyroscope, we estimate the rotation matrix over time $R(t)$ and use it to estimate the variation in the watch's gravity $g(t)$, in the watch's coordinate system. We sample acceleration $a(t)$ in the watch's coordinate system and it is polluted by gravity. Now we remove gravity and get $a_{rg}(t) = a(t) - g(t)$.

3. **Estimate $C(t)$ and calculate the projected acceleration.** Note that directly integrating $a_{rg}(t)$ has no physical meaning even if gravity has already been removed. This is because $a_{rg}(t)$ is along the watch's axes and the watch rotates overtime. Ideally, we want to integrate along the fixed directions of the absolute coordinate system. Those directions are $X(t)$, $Y(t)$ and $Z(t)$ mentioned before in $C(t)$. We can get $C(t)$ with the help of the rotation matrix $R(t)$. Thus, $a'_{rg}(t)$ can be obtained by projecting $a_{rg}(t)$ to $X(t)$, $Y(t)$ and $Z(t)$. Integrating $a'_{rg}(t)$ now yields the speed $v'(t)$, and integrating $v'(t)$ ultimately returns the displacement $s'(t)$.

4. **Calibrate by mean removal (speed and displacement).** Of course, this is erroneous, however, if we know that at time T, $v'(T) = 0$ and $s'(T) = 0$ (i.e., the watch has come to a stop), we can refine the estimates of speed and displacement by mean removal.

5. **Perform Kalman smoothing.** The displacement estimation is still not stable, and occasionally the result becomes poor. We carefully checked the data and detected that gravity estimation is not entirely reliable. Thus, we apply a Kalman smoothing to $a'_{rg}(t)$ to estimate the gravity estimation error $g'_e(t)$. The idea is to think about $a'_{rg}(t)$ as $g'_e(t)$ plus noise, where noise is generated due to the act of typing. We set a large noise parameter in Kalman smoothing such that the Kalman smoothing output does not closely follow $a'_{rg}(t)$, but it shows the underlying shifting trend

hidden in $a'_{rg}(t)$. Now, we refine $a'_{rg}(t)$ to $a'_{rg}(t) - g'_e(t)$. The performance improves consistently.

Figure 4.12 plots the final result comparison between *MoLe* and the original Android API method. It is clear that *MoLe* provides better watch displacement estimation.

4.4.2 Point Cloud Fitting

From the estimated displacements for each key, *MoLe* generates a *unlabeled point cloud (UPC)* for the attackee. Since the points in this UPC are not labeled, we fit the attacker's *character point cloud (CPC)* to the UPC. The key intuition is that the relative motions between keys (reflected in the relative locations of the point clouds) should bear similarity across all users. To achieve this fitting, we compute the convex hulls for the CPC and the UPC.

Observing that the fitting parameters for up and down hand displacements can be different, we compute two convex hulls for the CPC – one for all the positive X displacements (denoted by H^{CPC}_{pos}) and the other for all the negative X displacements (denoted by H^{CPC}_{neg}). Similarly, we compute two convex hulls for the UPC – H^{UPC}_{pos} and H^{UPC}_{neg}. We fit H^{CPC}_{pos} to H^{UPC}_{pos} and H^{CPC}_{neg} to H^{UPC}_{neg} respectively.

To fit a convex hull H_1 to another convex hull H_2, we first calculate their cords C_1 and C_2 which originate from the origin and pass through their centroids. Then, we (1) rotate H_1 such that C_1 aligns with C_2, (2) scale H_1 in the direction of C_1 such that C_1 equals to C_2, and (3) scale H_1 in the orthogonal direction of C_1 such that the area of H_1 equals to that of H_2. Figure 4.13 shows an example of point cloud fitting.

The metric for fitting is defined by the degree of overlap between the CPC and UPC convex hulls. More precisely, we compute the ratio of the intersection and union of the convex hulls.

An attacker might be able to generate multiple CPCs, perhaps from her accomplices. *MoLe* performs the fitting for each of these CPCs and selects the one that maximizes the intersection/union ratio. The rotated and scaled CPC

is now superimposed on the UPC and a framework is ready to estimate labels for each point in the UPC.

4.4.3 Bayesian Inference

Even if the keystroke detection and point cloud fitting are perfect, *MoLe* still does not know the characters typed by the right hand. Thus, as a first step, we attempt to fill in these "holes" to infer the complete word. The rather obvious step is to calculate the *posterior probability* of each word in the English dictionary, given the motion inferences from the left hand. The words corresponding to the top-K highest probabilities can be enumerated as candidates. We apply Bayes' theorem formulated as:

$$P(W \mid O) = \frac{P(O \mid W)P(W)}{P(O)} \tag{4.1}$$

where

- W is a candidate word from the dictionary and O is the observation motion data.
- $P(W|O)$ is the posterior probability of the word given the observed motion data.
- $P(O|W)$ is the likelihood function that estimates the probability of the word W based on the observed motion data.
- $P(W)$ is the prior probability which captures the word's occurrence frequency.
- $P(O)$ is the probability of the observation.

Since $P(O)$ is the same for all possible words, we are only interested in calculating $P(O|W)$ and $P(W)$. That is,

$$P(W \mid O) \quad \propto \quad P(O \mid W) \quad \times \quad P(W) \tag{4.2}$$

$P(W)$ can be obtained from a contemporary English corpus. In the current experiment, we assume $P(W)$ is equal among words, meaning each word has

the same occurrence frequency. The key goal translates to obtaining the maximum (or high) values of the likelihood $P(O|W)$. In the following, we present a few opportunities to refine the likelihood function and the posterior probability.

Step 1: Using the Number of Keystrokes Our first intuition is simply to employ the number of detected keystrokes as observations to match the word. The keystroke detector gives us the number of keys typed by the left hand and this number is used to match each word. For example, when two keys are detected, matching the word "the" produces higher likelihood than the word "teacher", because the number of peaks generated while typing "the" is much closer to two than "teacher". Now, for each of the detected keystroke, we would like to match them with the characters in the word. Since the keystroke could be caused by any characters in the word, we need to consider all possible assignments.

To calculate P(O | W), we can write

$$P(O \mid W) = P(N \mid W) = \sum_{(\alpha_1,...,\alpha_N)} P((c_{\alpha_1},...,c_{\alpha_N}) \mid W) \tag{4.3}$$

where N is the number of keystrokes and $(\alpha_1,...,\alpha_N)$ represents one possible N-element combination from $\{1,2,...,L\}$ and L is the word length. The summation adds up all possible combinations. Here, c_i is the ith character in W; $P((c_{\alpha_1},...,c_{\alpha_N}) \mid W)$ is the probability that N peaks are generated by $c_{\alpha_1},...,c_{\alpha_N}$.

For instance, let us assume two keystrokes are detected and we want to calculate the likelihood of the word "the" by using the keystroke detector result in Figure 4.10.

$$\begin{aligned} P(O=2 \mid W = "the") &= 0.986 * 0.035 * 0.119 \ (c_{\alpha_1} = t; c_{\alpha_2} = h) \\ &+ 0.986 * 0.965 * 0.881 \ (c_{\alpha_1} = t; c_{\alpha_2} = e) \\ &+ 0.014 * 0.035 * 0.881 \ (c_{\alpha_1} = h; c_{\alpha_2} = e) \\ &= 0.84 \end{aligned}$$

In the above equation, 0.119 means the probability that "e" is not detected and equals to $(1-0.881)$. In a similar way, we calculate the probability that "t" is not detected (0.014) and "h" is detected (0.035).

Thus, by iterating over all words in the dictionary, we can obtain the likelihood for each word. Of course, further refinements are possible.

Step 2: Consecutive Characters In some cases, our key-press timing module detects only one keystroke for two consecutive characters such as “er”, “sa” or “re”. These keys are adjacent on the keyboard, and the watch dips in so close succession that they are not separable. Therefore, we treat these character pairs as one key. Figure 4.14 shows the experimental results that these common character–pairs are detected as zero, one, or two key-presses. Evidently, treating them as a single key-press should be appropriate in a majority of the cases.

Step 3: Adding Watch Displacement *MoLe* is now ready to leverage the actual watch displacement. Assuming fingers are placed over the home position (“F” and “J”), recall that the key-press location estimation module computes the location of each key-press (Figure 4.4). Of course, the estimated location is not accurate due to the noise in the hardware, minute differences in the hand motions, minute differences in hand’s 2D orientation, etc. However, given that the CPC has been fitted to the user’s UPC, it is now possible to better predict the word by taking displacement into consideration. Thus, Equation 4.3 can be rewritten as:

$$\begin{aligned} & P(O \mid W) \\ & = P(N \cap d_i, i = 1,2,...,N \mid W) \\ & = \sum_{(\alpha_1,...,\alpha_N)} P((c_{\alpha_i},...,c_{\alpha_N}) \mid W)\, p((d_1,...,d_N) \mid (c_{\alpha_i},...,c_{\alpha_N}), W) \end{aligned} \tag{4.4}$$

where $p((d_1,...,d_N) \mid (c_{\alpha_i},...,c_{\alpha_N}), W)$ is probability density of typing $c_{\alpha_1},...,c_{\alpha_N}$ of W at character displacements $d_1,...,d_N$, and d_i is the ith character displacement in W.

MoLe models each character’s location as a Gaussian distribution. Assuming the distribution of displacement d_i only depends on current character c_{α_i}, we simplify Equation 4.4 as:

$$P(O \mid W) = \sum_{(\alpha_1,...,\alpha_N)} P((c_{\alpha_1},...,c_{\alpha_N}) \mid W) \prod_{i=1}^{N} p(d_i \mid c_{\alpha_i}) \tag{4.5}$$

where $p(d_i \mid c_{\alpha_i})$ is probability density of d_i given character c_{α_i}.

Step 4: Character Transitions We assumed above that each character displacement is independent. However, typing a word consists of sequential movements and the current displacement is indeed influenced by the location of the previous character. Figure 4.15 illustrates an example in which we compare the displacement of "a" when the previous character is "v" versus "r". Clearly, the distributions are different. For "ra", the displacement of typing "a" is shifted toward the position of character "r" because the little finger types "a" right after "r", before returning to home position. For "va", the displacement of "a" is clearly close to "v", because of the same reason.

Given this observation, we extend the likelihood function to the following:

$$P(O \mid W) \approx \sum_{(\alpha_1,\ldots,\alpha_N)} P((c_{\alpha_1},\ldots,c_{\alpha_N}) \mid W) \prod_{i=1}^{N} p(d_i \mid c_{\alpha_i}, c_{\alpha_{i-1}}) \tag{4.6}$$

Now the displacement probability density $p(d_i \mid c_{\alpha_i}, c_{\alpha_{i-1}})$ not only considers c_{α_i} but also the previous character $c_{\alpha_{i-1}}$.

Step 5: Keystroke Interval Timing of the key-presses on the left hand should also encode information about missing keys. We ask, given the time detected interval between consecutive keystrokes, what the probability is that there are N right-hand characters between them. Correct guesses of N can obviously help. For example, when typing the word "t h a n k s" (characters typed by the left hand are underlined), the observed interval between "t" and "a" may be expectedly shorter than between "a" and "s".

Figure 4.16 plots the distribution of time intervals for increasing lengths of character sequences. These sequences consists of right-hand characters in the middle and are surrounded by two left-hand characters. Unsurprisingly, the time interval generally increases with the number of keystrokes. However, we observe high variance. For example, even though the segment "-b i l i t-" and "-t i o n a-" both have three right-hand characters in the middle, the average interval of "-t i o n a-" is shorter than "-b i l i t-" due to hand geometry and typing familiarity.

Therefore, for better timing observation, we should obtain the time interval distribution of every possible character-sequence that is preceded and followed by two left-hand characters, and use this distribution in the Bayesian model.

The observation can be written into

$$\begin{aligned} & P(O \mid W) \\ & = P(N \cap d_i, i = 1,2,...,N \cap t_j, j = 1,2,...,N-1 \mid W) \\ & \approx \sum_{(\alpha_1,...,\alpha_N)} P((c_{\alpha_1},...,c_{\alpha_N}) \mid W) \prod_{i=1}^{N} p(d_i \mid c_{\alpha_i}, c_{\alpha_{i-1}}) p((t_1,...,t_{N-1}) \\ & \mid (c_{\alpha_1},...,c_{\alpha_N}),(d_1,...,d_N),W) \end{aligned} \tag{4.7}$$

where $N-1$ interval distributions, t_j with $j = 1,2,...,N-1$, are added into the observation. Note that the attacker and attackee are typing at slightly different speeds. *MoLe* compensates this speed bias with a factor k, calculated from the ratio of attacker to attackee's average typing interval.

4.5 Evaluation

4.5.1 Data Collection and Methodology

MoLe has been implemented on the Galaxy Gear Live smartwatch, which runs the latest Android Wear platform. When activated, the *MoLe* client on the watch continuously logs accelerometer and gyroscope readings at 200 Hz, along with timestamps. The sensor data is stored locally during data collection and transferred to the backend (MATLAB) server for analysis.

MoLe is evaluated with eight subjects, recruited by advertising about these experiments in the university campus. The subjects were offered an incentive of $10 per hour, and each subject invited to our lab for a two-hour session. All subjects were familiar with English typing (five are native English speakers, three are females). Each subject was asked to type 300 English words randomly selected from 5000 most frequently used words [89]. The word length ranged from 1 to 14, and was equally distributed. In total, we test 2400 words across all users.

Each subject was seated at a desk in front of a Lenovo laptop. Our experiment GUI popped up one word at a time on the laptop screen, and the volunteer's task was to type the same word in a text box on the screen. If any of the character is incorrectly typed, we discard the data and let the subject re-enter the word. Between each word recording, we ask the subjects to initialize their hand position on "F" and "J". During the typing, the laptop was also programmed to record the timing of the keystroke, which will later serve as ground truth. To collect the offline training data, two of the authors (pretending to be attackers) performed the same procedure, but with the top-500 longest words in the dictionary. Long words help capture the diversity of the typing patterns. The whole data collection is done on a Lenovo ThinkPad equipped with a regular full-sized keyboard.

For full ground truth recording, we mount an Android Samsung Galaxy S4 phone on top of the keyboard and use the front camera to capture the video of hand movement. We apply the camera calibration toolbox in Matlab [91] to calibrate the camera pixel, and measure the watch distance and location from each frame (Figure 4.1).

4.5.2 Performance Results

The following questions are of interest in this section:

- How well can *MoLe* guess each word (i.e., in an ordered list of guesses by *MoLe*, what is the rank of the actual word)?
- What factors affect this rank?
- How do different opportunities contribute toward overall performance?
- How can we prevent the threat introduced by *MoLe*?
- Does the type of keyboard matter?
- Can humans guess better by looking at a sequence of candidate guesses?

4.5.2.1 How Well Can *MoLe* Guess Each Word?

Figure 4.17 plots the CDF of rank, computed from each of the 2400 words typed by the subjects of the experiments. The average across all eight subjects is

shown in black, while *MoLe*'s performance for each individual subject shown in gray. We observe that the median rank of a word is 24, while for 30th percentile, the rank is 5. Put differently, when a user types a word, there is a 30% chance that *MoLe* would narrow down the typed word to only five possibilities, and a 50% chance to only 24 possibilities. Given that 5000 words are possible, this is an appreciable reduction of the search space, and makes it amenable to brute-force attacks.

Figure 4.18 plots the ranks of typed words for each test subject. The ranks are generally higher (i.e., *MoLe*'s guesses are better) for subjects S2, S5, and S8. On examining the video and sensor traces for these subjects, we find that they have lower variance in their hand movements, probably because they type as per the prescribed guidelines. We also test the case with perfect key-press detection (i.e., using the actual number and timing of keystrokes only from the left hand, gathered from the ground truth timing information recorded during the experiments). Surprisingly, the 30th percentile drops sharply to 1, meaning that *MoLe* can exactly guess the word; the 50th percentile drops to 6. Clearly, further improvements in key-press detection is the key to improving *MoLe*.

4.5.2.2 What Factors Affect the Rank?

Figure 4.19 plots the median rank of words for increasing word lengths. The rank generally decreases with word length greater than 6, primarily because (1) there is a greater number of keystrokes that gets detected in a longer word, and (2) because the number of words of that length reduces, in turn reducing the number of words it can be confused with. Words of length 4 to 7 on the other hand, have fewer keystrokes; also, there are many more words of such lengths, adding to the difficulty of detection.

Figure 4.20 shows the number of left-hand characters in a word. With an increasing number of left-hand characters, *MoLe* naturally gains richer information about the word, ultimately improving its ability to guess. When a word contains more than five left-hand characters, *MoLe* is able to bring down the rank below 20. When the number of left-hand characters is two to four, performance degrades because a large number of words have the same two-four left-hand characters in them.

4.5.2.3 Impact of Each Bayesian Opportunity

Section 4.4.3 leveraged a number of opportunities under the Bayesian model. Figure 4.21 shows the break-down of contributions from each of them. Using only the number of detected keystrokes (step 1 and 2 in Section 4.4.3), *MoLe* performs rather poorly on the dataset. When the displacement information is added (step 3 and 4), *MoLe* improves the rank from 340 to 49 at 50th percentile. Finally, when time interval is incorporated (step 5), the median rank improves from 49 to 24.

4.5.2.4 Impact of Sampling Rate

Figure 4.22 characterizes the impact of sensor sampling rate (200 Hz, 100 Hz, 50 Hz, 20 Hz) on the median rank of words for all of the subjects. Evidently, *MoLe*'s ability to guess degrades drastically with lower sampling rates – median ranking falls as 64, 141 and 1218. Perhaps this could be a way to mitigate the attack through smartwatches. A typing classifier could first detect whether the user is typing, and if so, the sampling rate of the sensor data can be diminished to less than 50 Hz.

4.5.2.5 Keyboard Variant

The difference in the shape of desktop and laptop keyboards may translate to decoding errors with *MoLe*. To test this, subject S5 was asked to repeat the same data collection process on a desktop keyboard. Figure 4.23 plots a histogram of the rank differences of each word, when decoded from the two keyboards. We notice that 45.2% of rank differences are less than 10. More specifically, the laptop keyboard presents a median rank of 10 and 30th percentile rank of 4. The computer keyboard's median rank is 14 and 30th percentile rank is still at 4. The results show that the system performance is close between two keyboards, *even though the attackers used the laptop keyboard for training the system.* If the attackers generated models from various keyboards, and applied the best one during the keystroke detector and cloud fitting process, the results can be even better.

4.5.2.6 Recovery via Human Observation

Although *MoLe* is not able to detect spaces and separate the words at this moment, we are interested to know how the threat would become even worse if this limitation is relaxed. To this end, we ask subject $S5$ to enter (one-by-one) the words from an actual sentence. Table 4.1 shows *MoLe*'s end-to-end prediction result for each of the words in the sentence (which contains eight words). For each word, the top-5 guesses are listed from top to down. As would be expected, the words in each column bear similarity in the character sequences embedded in them. For example, W_6 typically starts with "t" or "th" and W_8 contains many left-hand characters. We present this table to colleagues in our department and most of them could recover the sentence in a few minutes. We encourage the readers to reconstruct sentence on their own – the answer is made available at end of the Chapter 4.[2]

4.6 Points of Discussion

We discuss a few limitations and opportunities for improvement.

Confined to Separate Words *MoLe* is not yet a real-world attack since it is not able to detect the space bar and separate out words from a sentence. Also, without any priors on digit sequences, it is difficult to detect what digits users are typing. Additional work is necessary to further separate out these keystrokes. However, we believe there is opportunity. We have found early evidence that the magnetic field on the keyboard is quite telling of the position of the wrist. With some signal processing, the magnetic field may offer valuable hints on how the wrist is moving and when it is coming back to its original position. We leave this to future work.

Applying Nature Language Processing To recover the whole sentence, techniques from nature language processing (NLP) may also apply. For example, we can apply N-gram language models which predict the Nth word given a previous $N-1$ word sequence. Thus, even if a few words have low accuracy in the sentence, it may still be possible to infer the sentence, or even the broad semantic content.

Typing Activity Classifier In a real-world attack, we would first need to subject the sensor data to a classifier which will output whether the user is typing or not. Only when the user is known to be typing, should *MoLe* be applied on the data. We have not developed a "typing or not" detector in this chapter, but believe that it can be developed (perhaps from the orientation of the watch and the slight back-and-forth movement). Also, we need to be able to identify whether the watch is worn on the left or right hand, which remains for future work.

4.7 Related Work

We categorize this section into inferring keystrokes on traditional computer keyboards and sensor information leaks on smart devices.

Inferring Keystrokes on Computer Keyboards Many researches have attempted to infer keystrokes on computer keyboards. Various modalities have been leveraged, such as acoustics signal [92, 93], input timing analysis [94, 95, 96], RF radio [97], and electromagnetic emanations [98]. These researches have successfully delivered high-accuracy results. However, none of them use motion sensor data; also, to intercept physical signals and decode the data, these methods are typically required to install additional hardware or software, which make them somewhat difficult to widely deploy the attacks. In contrast, *MoLe* can be launched with ease on top of commericially available wearable devices. Marquardt et al. demonstrated the (Sp)iPhone [99] and showed that it is possible to use the accelerometer on an iPhone to recover text entered on a keyboard when the phone is placed nearby, on the same table surface. Both (Sp)iPhone and *MoLe* exploit side channels, however, *MoLe* uses the wrist motion data in 3D, which differs from the surface vibrations in (Sp)iPhone.

Sensor Information Leaks on Smart Devices Researchers have studied side channel attacks to infer keystrokes on smartphones and tablets [100, 101, 102, 103]. The core idea behind these works is that when typing on different locations on a virtual keyboard, the keystrokes cause distinct vibrations/rotations. The motion data on smartphones can thus be used to infer the tapped loca-

tion. TouchLogger [101] and accessory [100] are the early works, and use an accelerometer only to infer tap location on screen. Cai et al. [102] and Miluzzo et al. [103] advance the technique to infer keystrokes and show that a gyroscope has better accuracy than accelerometer-based inference. *MoLe* bears similarity in that it is also a side channel attack. However, there are two main differences. First, the above works are feature-based. They design features to capture distinct screen motions caused by the touches and train models with these features to infer the touch positions. In contrast, since smartwatch is on the user's wrist, we track the movement of the wrist to infer what the user has typed. Second, we are only able to sense partial keystrokes with one hand, as well as indirect data of the wrist. To recover the whole input word, *MoLe* must rely much more strongly on the Bayesian models. GyroPhone [104] presents a new type of threat to intercept human speech by using a gyroscope on a smartphone. The authors found that the MEMS gyro sensors are able to pick up air vibrations from sound at low frequency. *MoLe* is still different since it attempts to extract semantic understanding of the human's hand motions.

4.8 Conclusion

This chapter demonstrates that sensor data from smartwatches can leak information about what the user is typing on a regular (laptop or desktop) keyboard. By processing the accelerometer and gyroscope signals, tracking the wrist micro-motions, and combining them with the structure of valid English words, reasonable guesses can be made about typed words. Given the excitement around a smartwatch app store, such an attack can severely penetrate into the private lives of humans. While we find that diminishing the sampling rate of the accelerometer and gyroscope can alleviate the attack, we believe additional side channels like magnetic field variations need to be carefully investigated. Otherwise, wearable devices could soon become a "double-edged sword" slowing down future innovations on this platform.

4.9 Figures

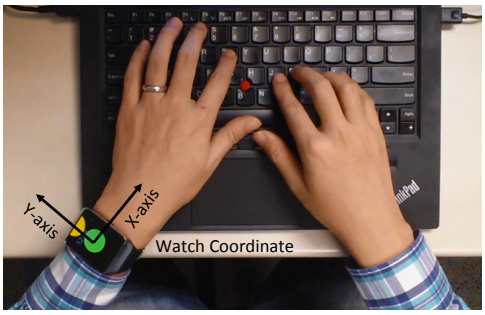


Figure 4.1: Watch coordinate system and ground truth measurement by recording hand typing with a smartphone camera view.

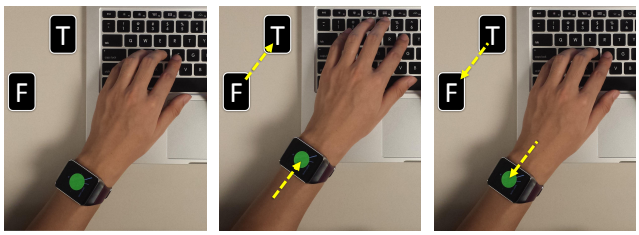


Figure 4.2: Three video frames show the process of typing “T” from “F”.

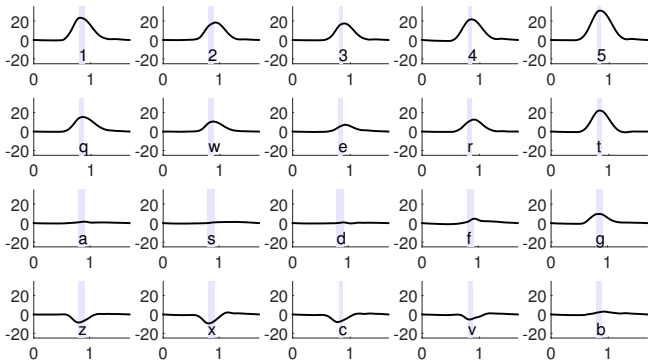


Figure 4.3: The watch X-axis displacements while a human types 20 characters. In the figures, X-axis is time in seconds and Y-axis is watch X-axis displacement in millimeter. The gray bar shows the keystroke press and release time interval.

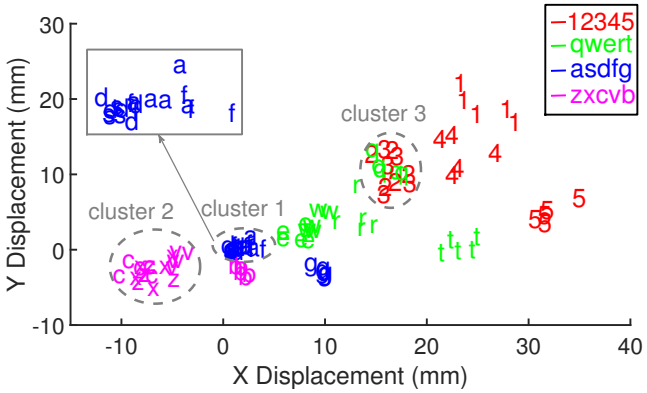


Figure 4.4: Watch 2D displacements while a human types 20 characters with the left hand. Each character is typed repeatedly five times. (0,0) is the initial location when left-hand fingers are placed on home position ("asdf"). Note that X-axis and Y-axis in the graph are in the watch's coordinate system.

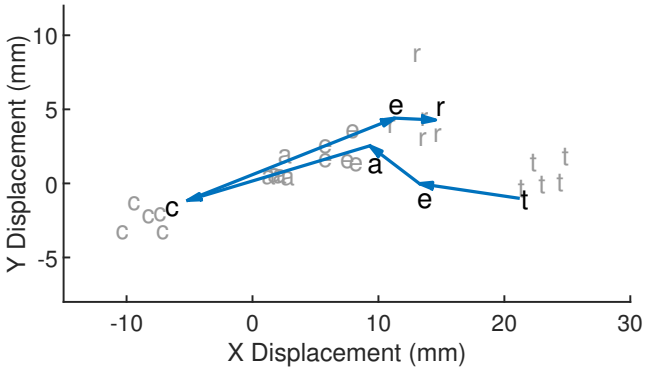


Figure 4.5: Comparison of typing “teacher” continuously (in black) against each character separately (in gray). Note that the positions of “e”, “a” and “c” are away from their original points due to sequential typing. Also, “h” is not captured due to right-hand typing.

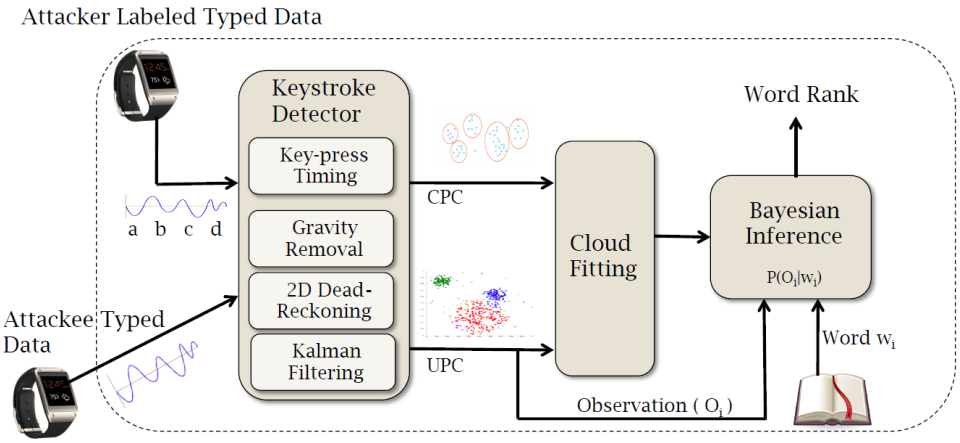


Figure 4.6: System overview: The typed data from users is pre-processed through gravity removal and timing analysis blocks, superimposed on the refitted typing templates, and passed through a Bayesian inference model that leverages the patterns and structures in English words to ultimately decode the typed words. Note that training is only required from the attacker's end; no training is needed for the user.

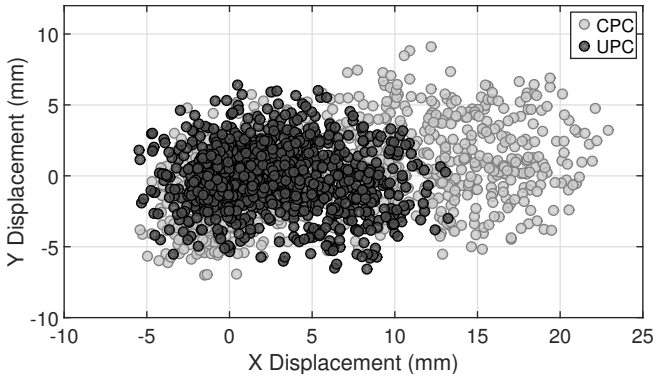


Figure 4.7: Character point cloud computed from attacker's data and unlabeled point cloud computed from user's data.

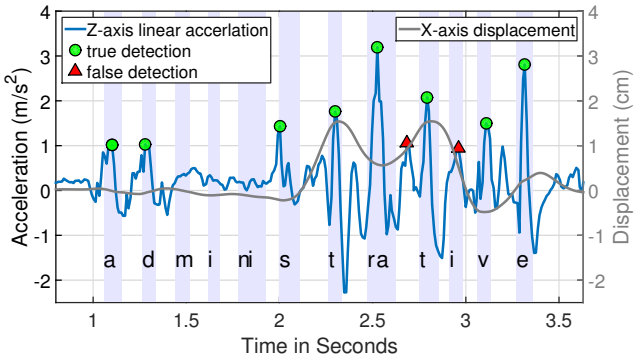


Figure 4.8: A simple peak detection scheme to detect keystrokes. The left Y-axis represents acceleration and the right Y-axis indicates displacement. Note that for “a”,“d”,“s” keystrokes, lower Z-axis acceleration is generated because of the left hand’s initial position. At time 2.7 and 3 seconds, there are two false detections due to the left hand moving from “a” to “t” and from “t” to “v”.

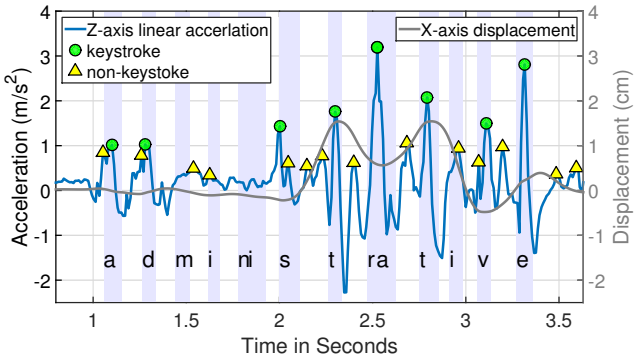


Figure 4.9: Bagged decision classification results: A peak detection tool with low thresholds is first applied to the Z-axis acceleration data and marks potential keystrokes (both yellow triangles and green circles). The classifier then identifies whether the peaks are keystrokes or not. Note that for the first "a" and "d", since two peaks are too close, the classifier would identify only one peak with the highest Z-axis acceleration within a time window.

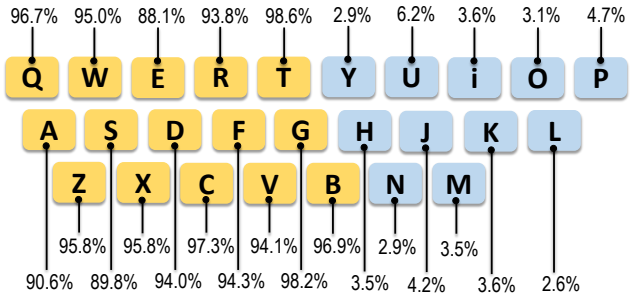


Figure 4.10: Keystroke detection rate for each character.

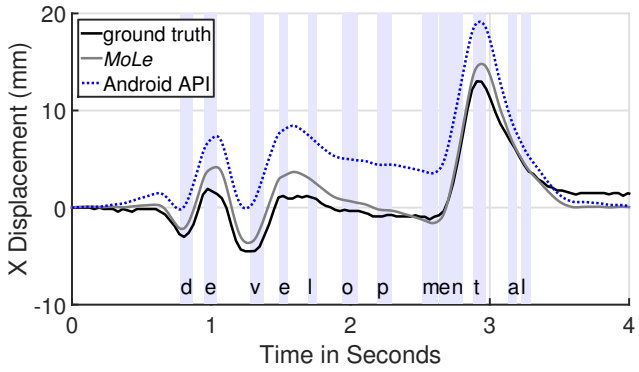


Figure 4.11: Comparison of *MoLe* and Android API X-axis displacement results. The Y-axis has similar results but is omitted due to space.

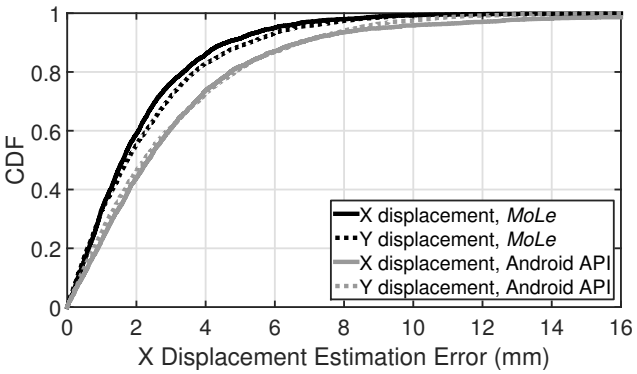


Figure 4.12: Comparison of *MoLe* and Android API displacement results.

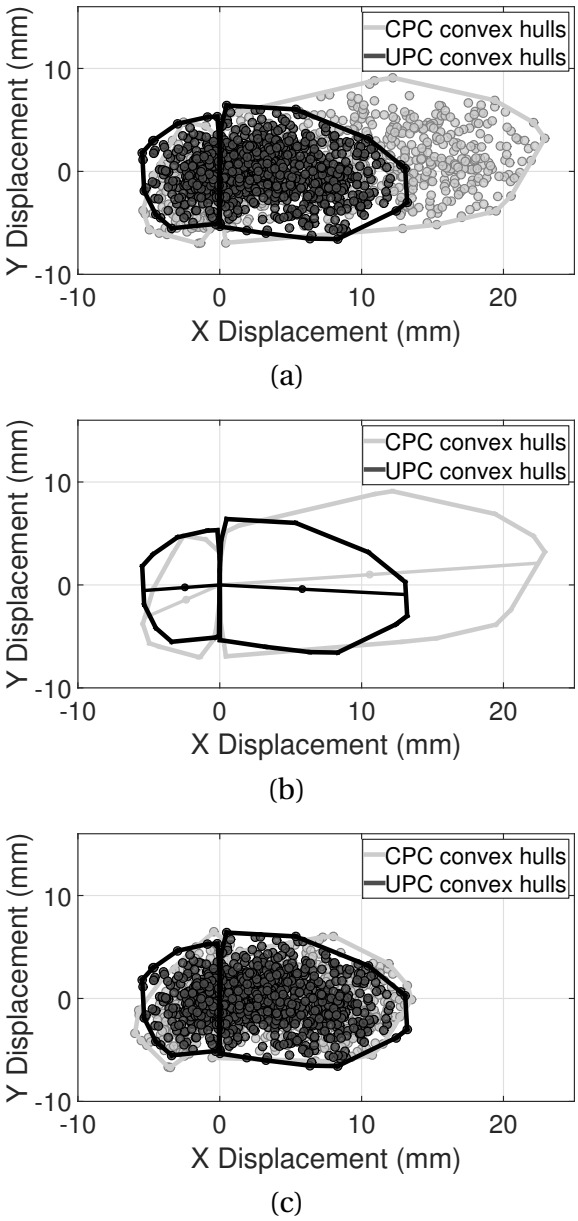


Figure 4.13: Point cloud fitting. Black points are the CPC attacker template and gray points are UPC from the attackee. (a) Find each convex hull. (b) Calculate the centroids and perform rotating and scaling. (c) Point cloud fitting result.

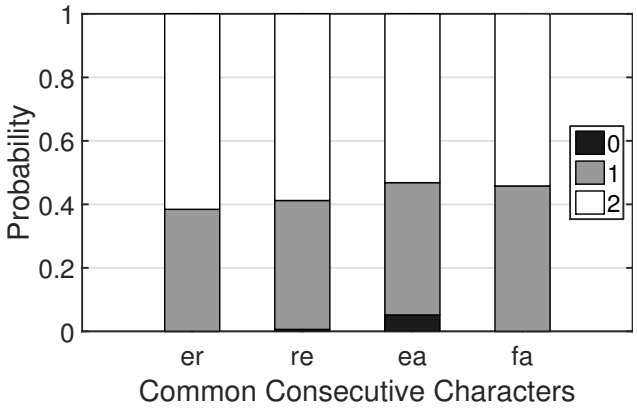


Figure 4.14: The probability of number of keystroke detections for consecutive characters.

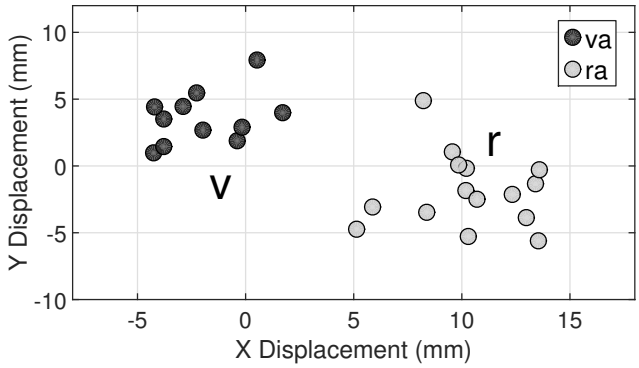


Figure 4.15: Comparison of “a” displacement while the previous character is “v” or “r”. The key locations of v and r are marked in the figure.

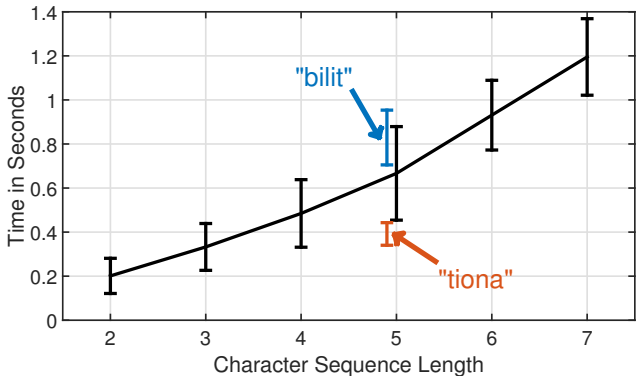


Figure 4.16: Y-axis is the time interval between two detected keystrokes and X-axis is the number of sequence length.

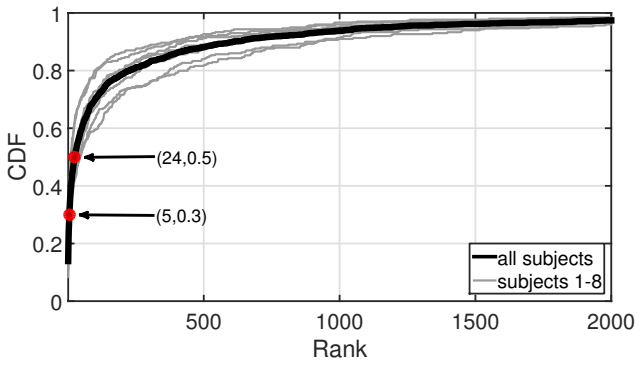


Figure 4.17: CDF of rank computed for each of the 2400 words typed by eight subjects.

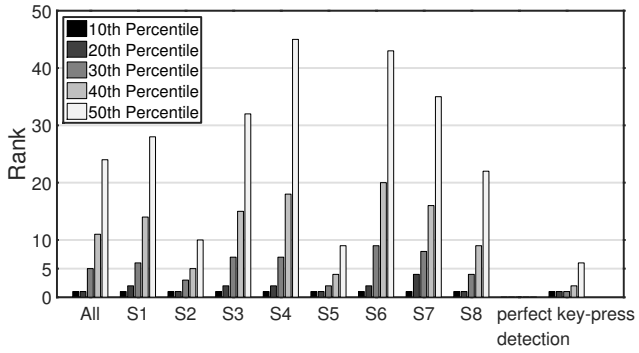


Figure 4.18: Rank of average, across users and with perfect key-press detection.

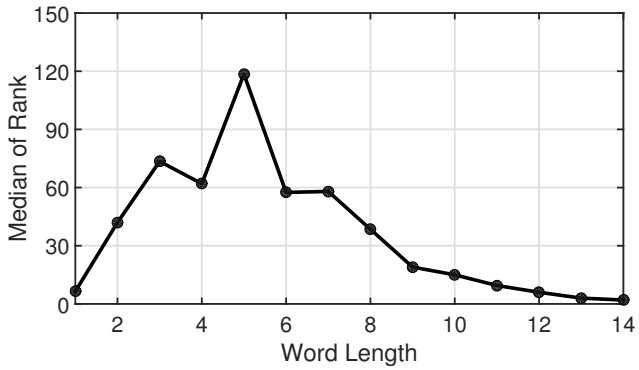


Figure 4.19: Median rank plotted against increasing word length – words of length 4 to 7 show the worst performance due to fewer keys to be detected while such words occur in large numbers.

Figure 4.20: Variation of rank against the number of characters typed by the left hand.

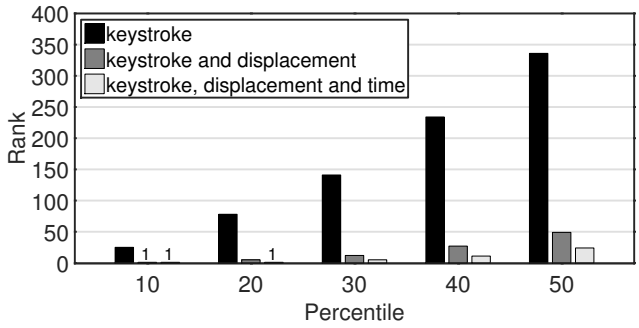


Figure 4.21: Contribution of different opportunities toward the overall performance of *MoLe*.

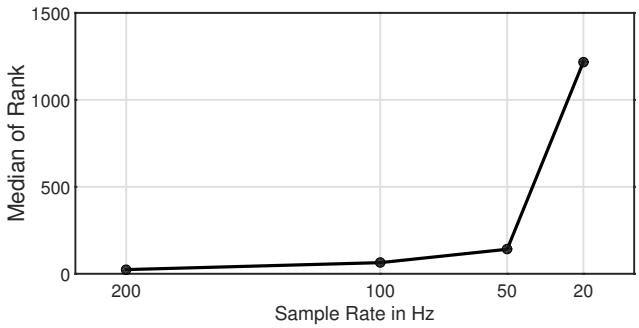


Figure 4.22: Lower sensor sampling rate rapidly reduces the ability to guess the word, perhaps indicating a way to thwart *MoLe*'s attack.

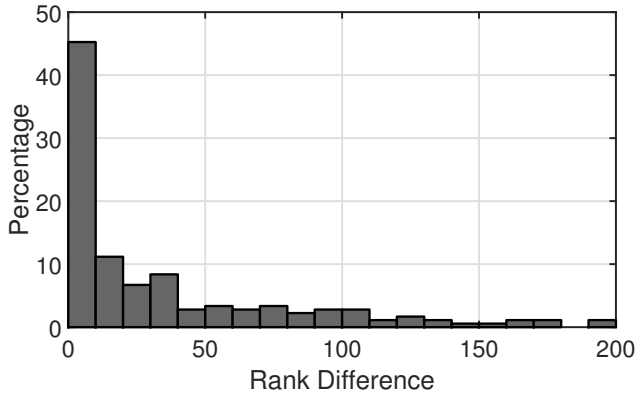


Figure 4.23: A histogram shows the rank difference of each word between two keyboards. The trend indicates that most words have a similar rank, so the difference is low.

4.10 Tables

Table 4.1: *Can you guess the correct sentence?* The words (W_1-W_8) in each column are ranked in decreasing order of probability. Note that some words may not feature in the top-5 words presented in each column. The answer is made available at end of the Chapter 4.[2]

Rank	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
1.	motor	pistol	profound	technology	angel	those	that	disappear
2.	monitor	list	journalism	remaining	spray	today	tight	discourse
3.	them	but	originally	telephone	super	third	tightly	secondary
4.	the	lost	original	meanwhile	fire	through	thirty	adviser
5.	then	most	profile	headline	shore	towel	truth	discover

[2] "The most profound technologies are those that disappear" - Mark Weiser, 1991

CHAPTER 5

CONCLUSION

Mobile devices are becoming a sensing and computing lens for society. Human behavior manifests in these sensing dimensions and we think it is possible to measure human behavior through motion data processing. Our research has focused on designing sensing and inferencing techniques, with an emphasis on location sensing and tracking.

We investigated the benefits of fusing smartphone sensors into the indoor localization framework. Results from UnLoc suggest indoor environments are rich in landmarks. These landmarks can be sensed by smartphones and located using noisy sensor data, ultimately improving localization performance. Continuing with the goal of indoor localization, we proposed a different point in the design landscape, VideoLoc, where feeds from surveillance cameras were integrated with smartphone sensors to offer highly precise localization. We also conducted research on micro-scale tracking of the human wrist and demonstrated that motion data processing is a "double-edged sword" – it offers value but also reveals incredible details about people's lives.

In the future, we plan to spend several more years on this theme, but broadening to various other sensing dimensions and applications. More specifically, we plan to do both bottom-up and top-down research. In the bottom-up research, we plan to develop deeper inferencing techniques beyond motion sensors and include cameras, sound, and wireless signals, in a way that the whole is greater than the sum of parts. In a second thread of top-down research, we plan to build systems and applications that utilize these inference techniques. Our current interests are in mobile health, smart cities, vehicular analytics and others, essentially trying to expose human behavior to various decision-making processes. Take mobile health as an example. The load on the healthcare system is growing tremendously. Sensing and inference technology, situated inside the home or on the body, would have to be harnessed to perform triage.

To this end, we will focus on various forms of activity and gesture recognition, as well as “change point detection” algorithms to identify when the important medical changes are occurring. We believe that the healthcare system of tomorrow will have to develop a bridge between physical hospitals and the context of the home – our research will help in building this bridge.

REFERENCES

[1] "Quantified self," http://quantifiedself.com/.

[2] "Fitbit," http://www.fitbit.com/.

[3] "Apple health," http://www.apple.com/ios/health/.

[4] "Microsoft health," https://www.microsoft.com/microsoft-health/en-us.

[5] "Android fitness," http://www.androidfitness.net/apps-pedometer.

[6] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. ACM, 2012, pp. 197–210, doi:10.1145/2 307 636.2 307 655.

[7] H. Wang, S. Sen, A. Mariakakis, A. Elgohary, M. Farid, M. Youssef, and R. Roy Choudhury, "Video: Unsupervised indoor localization (unloc): Beyond the prototype," https://www.youtube.com/watch?v=a35rCtUDg10.

[8] H. Wang, X. Bao, R. R. Choudhury, and S. Nelakuditi, "Insight: Recognizing humans without face recognition," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '13. ACM, 2013, pp. 7:1–7:6.

[9] H. Wang, X. Bao, R. Roy Choudhury, and S. Nelakuditi, "Visually fingerprinting humans without face recognition," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15. ACM, 2015, pp. 345–358, doi:10.1145/2 742 647.2 742 671.

[10] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. ACM, 2015, pp. 155–166, doi:10.1145/2 789 168.2 790 121.

[11] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "Mole video," https://www.youtube.com/watch?v=scZEHExzems.

[12] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *INFOCOM, 2000 Proceedings IEEE*, vol. 2, 2000, pp. 775–784.

[13] I. Constandache, R. R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[14] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. ACM, 2005, pp. 205–218.

[15] P. Krishnan, A. S. Krishnakumar, W.-H. Ju, C. Mallows, and S. N. Gamt, "A system for LEASE: Location estimation assisted by stationary emitters for indoor RF wireless networks," in *INFOCOM, 2004 Proceedings IEEE*, vol. 2, 2004, pp. 1001–1011.

[16] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, "Accuracy characterization for metropolitan-scale WiFi localization," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. ACM, 2005, pp. 233–245.

[17] W. G. Griswold, P. Shanahan, S. W. Brown, R. Boyer, M. Ratto, R. B. Shapiro, and T. M. Truong, "ActiveCampus: Experiments in community-oriented ubiquitous computing," *Computer*, vol. 37, no. 10, pp. 73–81, 2004.

[18] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "PinPoint: An asynchronous time-based location determination system," in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '06. ACM, 2006, pp. 165–176.

[19] R. J. Fontana and S. J. Gunderson, "Ultra-wideband precision asset location system," in *Ultra Wideband Systems and Technologies, 2002. Digest of Papers. 2002 IEEE Conference on*, 2002, pp. 147–150.

[20] D. Niculescu and B. Nath, "VOR base stations for indoor 802.11 positioning," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '04. ACM, 2004, pp. 58–69.

[21] N. B. Priyantha, "The cricket indoor location system," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[22] A. Savvides, C.-C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '01. ACM, 2001, pp. 166–179.

[23] J. Xiong and K. Jamieson, "SecureAngle: Improving wireless security using angle-of-arrival information," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. ACM, 2010, pp. 11:1–11:6.

[24] S. Sen, R. R. Choudhury, B. Radunovic, and T. Minka, "Precise indoor localization using PHY layer information," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ser. HotNets-X. ACM, 2011, pp. 18:1–18:6.

[25] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: Mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '09. ACM, 2009, pp. 261–272.

[26] S. Kumar and D. Shepherd, "SensIT: Sensor information technology for the warfighter," in *Proceedings of the 4th International Conference on Information Fusion*, 2001, pp. 1–7.

[27] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: Human localization using mobile phones," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10. ACM, 2010, pp. 149–160.

[28] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*. Springer, 2004, pp. 1–17.

[29] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. ACM, 2011, pp. 141–154.

[30] F. Evennou and F. Marx, "Advanced integration of WIFI and inertial navigation systems for indoor mobile positioning," *EURASIP Journal on Applied Signal Processing*, pp. 164–164, 2006.

[31] "Step size in pedometers," http://walking.about.com/cs/pedometers/a/pedometerset.htm.

[32] M. Alzantot and M. Youssef, "UPTIME: Ubiquitous pedestrian tracking using mobile phones," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 3204–3209.

[33] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10. ACM, 2010, pp. 173–184.

[34] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, ser. UbiComp '08. ACM, 2008, pp. 114–123.

[35] M. Youssef, M. A. Yosef, and M. El-Derini, "GAC: Energy-efficient hybrid GPS-accelerometer-compass GSM localization," in *Global Telecommunications Conference, 2010 IEEE*, 2010, pp. 1–5.

[36] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Eighteenth National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, 2002, pp. 593–598.

[37] B. Ferris, D. Fox, and N. D. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, vol. 7, no. 1, 2007, pp. 2480–2485.

[38] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor localization without infrastructure using the acoustic background spectrum," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. ACM, 2011, pp. 155–168.

[39] J. Lester, T. Choudhury, and G. Borriello, "A practical approach to recognizing physical activities," in *Pervasive Computing*. Springer, 2006, pp. 1–16.

[40] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Real-time detection and tracking for augmented reality on mobile phones," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 355–368, 2010.

[41] P. Mistry and P. Maes, "SixthSense: A wearable gestural interface," in *ACM SIGGRAPH ASIA 2009 Sketches*, ser. SIGGRAPH ASIA '09. ACM, 2009, p. 11:1.

[42] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, 2003.

[43] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. CVPR 1991. IEEE Conference on*, 1991, pp. 586–591.

[44] K. W. Bowyer, "Face recognition technology: Security versus privacy," *IEEE Technology and Society Magazine*, vol. 23, no. 1, pp. 9–19, 2004.

[45] C. Bo, G. Shen, J. Liu, X.-Y. Li, Y. Zhang, and F. Zhao, "Privacy.tag: Privacy concern expressed and respected," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '14. ACM, 2014, pp. 163–176.

[46] H. Wang, Z. Wang, G. Shen, F. Li, S. Han, and F. Zhao, "Wheelloc: Enabling continuous location service on mobile phone for outdoor scenarios," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 2733–2741.

[47] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part II*, ser. ECCV'12. Springer-Verlag, 2012, pp. 645–659.

[48] P. Dollár, "Piotr's Image and Video Matlab Toolbox (PMT)," http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.

[49] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[50] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1820–1833, 2011.

[51] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005, pp. 65–72.

[52] M. Eichner and V. Ferrari, "Calvin upper-body detector," http://groups.inf.ed.ac.uk/calvin/calvin_upperbody_detector/.

[53] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *Computer Vision and Pattern Recognition, 2011. CVPR 2011. IEEE Conference on*, 2011, pp. 1385–1392.

[54] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[55] S. T. Birchfield and S. Rangarajan, "Spatiograms versus histograms for region-based tracking," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Conference on*, vol. 2, 2005, pp. 1158–1163.

[56] C. O. Conaire, N. E. O'Connor, and A. F. Smeaton, "An improved spatiogram similarity measure for robust object localisation," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 1, 2007, pp. I–1069–I–1072.

[57] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. ACM, 2010, pp. 49–62.

[58] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[59] N. Robertson and I. Reid, "A general method for human activity recognition in video," *Computer Vision and Image Understanding*, vol. 104, no. 2, pp. 232–248, 2006.

[60] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*. Springer, 2004, pp. 1–17.

[61] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi, "TagSense: A smartphone-based approach to automatic image tagging," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. ACM, 2011, pp. 1–14.

[62] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, 2004.

[63] J. Man and B. Bhanu, "Individual recognition using gait energy image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 2, pp. 316–322, 2006.

[64] D. Jung, T. Teixeira, and A. Savvides, "Towards cooperative localization of wearable sensors using accelerometers and cameras," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[65] A. Ashok, M. Gruteser, N. Mandayam, J. Silva, M. Varga, and K. Dana, "Challenge: Mobile optical networks through visual MIMO," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10. ACM, 2010, pp. 105–112.

[66] A. Mayberry, P. Hu, B. Marlin, C. Salthouse, and D. Ganesan, "iShadow: Design of a wearable, real-time mobile gaze tracker," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14. ACM, 2014, pp. 82–94.

[67] R. Tenmoku, M. Kanbara, and N. Yokoya, "A wearable augmented reality system using positioning infrastructures and a pedometer," in *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, 2003, pp. 110–117.

[68] A. Henrysson and M. Ollila, "UMAR: Ubiquitous mobile augmented reality," in *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '04. ACM, 2004, pp. 41–45.

[69] W. Piekarski and B. Thomas, "ARQuake: The outdoor augmented reality gaming system," *Communications of the ACM*, vol. 45, no. 1, pp. 36–38, 2002.

[70] J. Rekimoto and Y. Ayatsuka, "CyberCode: Designing augmented reality environments with visual tags," in *Proceedings of DARE 2000 on Designing Augmented Reality Environments*, ser. DARE '00. ACM, 2000, pp. 1–10.

[71] "Qualcomm Vuforia," https://www.qualcomm.com/products/vuforia.

[72] "Videoguide, Antoni Gaudi Modernist Museum in Barcelona," http://www.casabatllo.es/en/visit/videoguide.

[73] N. Raval, A. Srivastava, K. Lebeck, L. Cox, and A. Machanavajjhala, "MarkIt: Privacy markers for protecting visual secrets," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ser. UbiComp '14 Adjunct. ACM, 2014, pp. 1289–1295.

[74] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 788–801.

[75] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 247–266, 2007.

[76] B. Yang and R. Nevatia, "Online learned discriminative part-based appearance models for multi-human tracking," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 484–498.

[77] C. H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*, June 2010, pp. 685–692.

[78] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video," in *Computer Vision and Pattern Recognition, 2011. CVPR 2011. IEEE Conference on*, June 2011, pp. 3457–3464.

[79] Z. Li, Q. L. Tang, and N. Sang, "Improved mean shift algorithm for occlusion pedestrian tracking," *Electronics Letters*, vol. 44, no. 10, pp. 622–623, May 2008.

[80] V. K. Singh, B. Wu, and R. Nevatia, "Pedestrian tracking by associating tracklets using detection residuals," in *Motion and Video Computing, 2008. WMVC 2008. IEEE Workshop on*, Jan 2008, pp. 1–8.

[81] S. Khan, O. Javed, Z. Rasheed, and M. Shah, "Human tracking in multiple cameras," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, 2001, pp. 331–336.

[82] A. Nakazawa, H. Kato, and S. Inokuchi, "Human tracking using distributed vision systems," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, Aug 1998, pp. 593–596.

[83] S. Agrawal, I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves, and F. DeRuyter, "Using mobile phones to write in air," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. ACM, 2011, pp. 15–28.

[84] S. Nirjon, J. Gummeson, D. Gelb, and K.-H. Kim, "TypingRing: A wearable ring platform for text input," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15. ACM, 2015, pp. 227–239.

[85] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, 2009, pp. 1–9.

[86] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '15. ACM, 2015, pp. 9–14.

[87] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis, "RisQ: Recognizing smoking gestures with inertial sensors on a wristband," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '14. ACM, 2014, pp. 149–161.

[88] S. Yun, Y.-C. Chen, and L. Qiu, "Turning a mobile device into a mouse in the air," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15. ACM, 2015, pp. 15–29.

[89] "Word Frequency Dataset," *http://www.wordfrequency.info/*.

[90] "MATLAB Peak Analysis Library," *http://www.mathworks.com/help/signal/examples/peak-analysis.html*.

[91] "Camera Calibration Toolbox for Matlab," *http://www.vision.caltech.edu/bouguetj/calib_doc/*.

[92] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Transactions on Information and System Security*, vol. 13, no. 1, pp. 3:1–3:26, 2009.

[93] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, 2004, pp. 3–11.

[94] D. Foo Kune and Y. Kim, "Timing attacks on PIN input devices," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. ACM, 2010, pp. 678–680.

[95] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, ser. SSYM'01. USENIX Association, 2001.

[96] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, 2009, pp. 125–134.

[97] "Key Sweeper," *http://samy.pl/keysweeper/*.

[98] M. Vuagnoux and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards," in *Proceedings of the 18th Conference on USENIX Security Symposium*, ser. SSYM'09. USENIX Association, 2009, pp. 1–16.

[99] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(Sp)iPhone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. ACM, 2011, pp. 551–562.

[100] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: Password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '12. ACM, 2012, pp. 9:1–9:6.

[101] L. Cai and H. Chen, "TouchLogger: Inferring keystrokes on touch screen from smartphone motion," in *Proceedings of the 6th USENIX Conference on Hot Topics in Security*, ser. HotSec'11. USENIX Association, 2011, pp. 9–9.

[102] L. Cai and H. Chen, "On the practicality of motion based keystroke inference attack," in *Proceedings of the 5th International Conference on Trust and Trustworthy Computing*, ser. TRUST'12. Springer-Verlag, 2012, pp. 273–290.

[103] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: Your finger taps have fingerprints," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. ACM, 2012, pp. 323–336.

[104] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, 2014, pp. 1053–1067.