DISS. ETH NO. 24267

Mobile Sensing: GPS Localization, WiFi Mapping, Applications, and Risks

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

PASCAL BISSIG

M. Sc., ETH Zurich, Switzerland

born on 15.07.1987

citizen of
Switzerland

accepted on the recommendation of

Prof. Dr. Roger Wattenhofer, examiner
Prof. Dr. Jie Liu, co-examiner

2017

Abstract

Localization is a fundamental feature of mobile systems such as smartphones, airplanes or self-driving cars. Albeit current smartphones include GPS receivers, accelerometers, and gyroscopes, localization is still difficult in certain environments such as indoors. First we show how WiFi signal strength measurements and motion data recorded with smartphones can be used to create accurate signal strength maps. The maps are created from data collected by a user walking around with a smartphone in the trouser pocket. From this data, we can accurately observe how the person moves by linking sensor measurements to how people walk. Exploiting the signal strength distributions recorded along the way, we show how the error that aggregates from inaccuracies in the motion estimation can be reduced. The signal strength maps are useful for localization indoors when GPS is unavailable.

In Chapter 3 we introduce a GPS receiver design which allows for aggressive duty cycling. This means that the RF front end only has to be turned on for a very short time to collect enough data for localization. From the short signal recordings we compute accurate position fixes efficiently even if the uncertainty on position and time are larger than what can be expected from synchronization and localization using cell towers. The receiver design can be tailored for different applications. For example, short signal recordings can be stored without computing a position right away. This allows for low power and long term tracking as well as instant position annotation for photos or other data. Alternatively, the short signal duration allows for fast initial position estimation for connected devices.

Chapters 4 and 5 are focused on device motion rather than device location. We discuss how motion data can be used to infer personal data such as pins or passwords. Our large scale user study reveals that, in uncontrolled environments, touch input on smartphone screens can be inferred from the same devices' motion data. We also show how motion data can be used constructively to recognize gestures and sequences thereof. For gesture recognition, we use motion data from people writing on whiteboards wearing smartwatches. Albeit the high recognition accuracy of isolated letters, sequences of letters are hard to segment. The built in microphone picks up the sounds caused by the pen which can help to segment the input into individual letters.

In the two final chapters, we localize sound sources in the environment of smartphones. This can be especially useful for discussion analysis or discussion diarisation. We first discuss how a single device can be used to deliver basic functionality. Adding more devices that collaborate, we arrive

at a system that works in most real situations, overcoming clock inaccuracies and device heterogeneity.

Zusammenfassung

Lokalisierung ist in vielen Geräten und Situationen nützlich, wie zum Beispiel in Smartphones, Flugzeugen oder selbstfahrenden Autos. Smartphones sind oft mit GPS Receivern, Beschleunigungssensoren und Gyroskopen ausgestattet. Dennoch ist es in Umgebungen wie in Gebäuden schwierig, solche Geräte genau zu lokalisieren.

Im ersten Kapitel zeigen wir, wie Messungen der WiFi-Signalstärke und Bewegungsdaten, die mit Smartphones aufgezeichnet wurden, verwendet werden können, um Signalstärkeverteilungen zu kartografieren. Die dazu nötigen Daten können mit einem Smartphone aufgezeichnet werden, welches in der Hosentasche getragen wird. Die Bewegung der tragenden Person kann genau verfolgt werden, wenn ein Smartphone in der Hosentasche getragen wird. Gleichzeitig können die Signalstärken der sichtbaren WiFi-Infrastruktur aufgezeichnet werden. Diese verwenden wir, um sich akkumulierende Ungenauigkeiten in der Bewegungsmessung zu reduzieren. Die resultierenden räumlichen Signalstärkeverteilungen sind nützlich für die Lokalisierung in Räumen, die keinen GPS-Empfang zulassen.

In Kapitel 3 stellen wir einen GPS-Empfänger der kurze Arbeitszyklen zulässt. Dies bedeutet, dass der Empfänger lediglich sehr kurze Signalaufzeichnungen benötigt, um eine exakte Position berechnen zu können. Dies ist auch dann möglich, wenn die wirkliche Position nur bis auf einige hundert Kilometer genau bekannt ist. Solch eine ungenaue Positionsschätzung ist durch Mobilfunk oder die letzte bekannte Position einfach zu erreichen. Der Empfänger kann für verschiedene Anwendungen angepasst werden. Zum Beispiel können die sehr kurzen Signalaufzeichnungen direkt gespeichert werden, ohne eine Position zu errechnen. Dies erlaubt es uns, mit wenig Energie für lange Zeit die Position des Empfängers zu verfolgen oder Fotos sofort mit Positionsdaten zu annotieren. Alternativ dazu kann der Empfänger die Zeit für die erste Positionsberechnung herkömmlicher GPS-Empfänger verkürzen, falls eine Netzwerkverbindung vorhanden ist.

In Kapitel 4 und 5 behandeln wir, wie die Bewegung eines Smartphones verwendet werden kann, um auf persönliche Daten wie Pins oder Passwörter zurückzuschliessen. Unsere Benutzerstudie zeigt, wie in unkontrollierten Umgebungen Eingaben auf Touchscreens aus den entsprechenden Bewegungsdaten abgeleitet werden können. Bewegungsdaten können auch konstruktiv verwendet werden, um Benutzergesten zu erkennen. Für die Gestenerkennung benutzen wir Bewegungsdaten, die mit Smartwatches aufgezeichnet werden, während eine Person an ein Whiteboard schreibt. Obwohl die Erkennung einzelner Buchstaben gute Ergebnisse liefert, ist es schwierig, Sequenzen von Buchstaben korrekt zu segmentieren. Audiodaten helfen dabei,

sequenzen von Buchstaben zu unterteilen und dadurch die Segmentierung zu vereinfachen.

In den zwei letzten Kapiteln lokalisieren wir Geräuschquellen in der Umgebung von Smartphones. Dies ist besonders nützlich, um Diskussionen zu analysieren. Wir zeigen zuerst, wie ein einzelnes Gerät diese Funktionalität bereitstellen kann. Wenn wir mehrere kooperierende Geräte verwenden, können wir in den meisten realistischen Situationen gute Ergebnisse erzielen. Wir präsentieren dazu Lösungen der praktischen Probleme wie Ungenauigkeiten der Uhren und Geräteinhomogenität.

Acknowledgements

When spending years doing a PhD that, by design, consists of a large number of smaller projects, some failure is unavoidable. Even though most of my projects failed, I was lucky enough to be surrounded by people who helped me to finally arrive at this - a thesis of a few small projects that worked out more or less the way I hoped.

First, I thank my supervisor Roger Wattenhofer for giving me the opportunity to write this thesis in his group. His steady advice kept many projects on track even though things turned in unexpected directions. Even small comments, probably though up in the moment, often greatly and positively affected the outcome of many projects.

Then, I thank my co-referee Jie Liu for taking the time to review this thesis and for serving on my committee.

There are many more people who made my time in the Distributed Computing Group a delightful experience. I thank (in alphabetical order) my co-workers:

Barbara Keller, for believing me that chauvinist jokes are really just jokes. Beat Futterknecht, for solving countless administrative problems and knowing what signatures are good for. Benny Gächter, for showing me that not all sysadmins are from hell. Christian Decker, for showing that you don't need to be a Bitcoin millionaire. Darya Melnyk, for keeping Philipp sane and always being there (literally). David Stolz, for teaching me how to quit so it does not look like giving up. Friederike Brütsch, for being patient with my ineptitude towards anything financial. Georg Bachmeier, for being the happiest, saddest, optimistic, cynical wannabe lawyer on the block. Gino Brunner, for getting through countless hours of student meetings with me. Jara Uitto, for being a climbing buddy making a supreme case for Finnish people. Jochen Seidel, for showing how to truly live opportunistically. Klaus Förster, for being the kindest guy ever while having the nastiest jokes ready at all times. Laura Peer, for giving me a new home and making my gadget-addiction look like a fleeting hobby. Manuel Eichelberger, for sticking with that weird GPS idea without any sign of slowing down. Michael König, for showing that being smart is not an option for me. Pankaj Khanchandani, for being surprised about nasty jokes yet having the nastiest comebacks himself. Philipp Brandes, for working hard but surfing harder. Samuel Welten, for getting me into this and his ever positive attitude towards everything. Sebastian Brandt, for showing that apparently you can improve your tabletop soccer skills by playing chess. Simon Tanner, for his incredibly efficient and pragmatic way of crushing problems. Thomas Ulrich, for showing me that I truly am getting old. Tobias Langner, for insisting that wine can be, and actually is, corked. Yuval Emek, for getting us all through a year without Roger being around. Yuyi Wang, for showing that one can do this twice.

Last but not least, I thank the people most important in my life: My parents, Joe and Trudy, and my sister, who supported my weird choices in career and hobbies alike; my girlfriend Katja, whose affection helped me to stay sane in moments of doubt while amplifying the anticipation and moments of joy; and my friends Tobi and Peter, with whom I shared disjoint yet great adventures through the years.

Contents

1 Introduction **1**
1.1 Collaborations and Contributions 4

2 WiFi Indoor Localization **7**
2.1 Related Work 8
2.2 Motion 11
2.3 Observation 14
2.4 Fusion 15
2.5 Localization 17
2.6 Experiments and Results 17
2.7 Conclusion 21

3 Fast and Robust GPS Localization **25**
3.1 Related Work 27
3.2 GPS Fundamentals 28
3.3 Localization Method 32
3.4 Evaluation 41
3.5 Conclusion 47

4 Touch in Motion **51**
4.1 Related Work 52
4.2 Method 53
4.3 Data Set 57
4.4 Evaluation 57
4.5 Additional Observations 66
4.6 Conclusion 67

5 Text in Motion **71**
5.1 Related Work 72
5.2 Method 76
5.3 Evaluation 79
5.4 Conclusion 84

6 Discussion Diarisation **87**
6.1 Related Work 88
6.2 Measurement System 90
6.3 Application 93
6.4 Evaluation 94
6.5 Conclusion 95

7 Distributed Discussion Diarisation **99**
7.1 Related Work 100
7.2 Method 101
7.3 Evaluation 108
7.4 Android Application 110
7.5 Conclusion 111

8 Conclusion **113**

1 Introduction

“See first, think later, then test. But always see first. Otherwise, you will only see what you were expecting. Most scientists forget that.”

Douglas Adams

Douglas Adams highlights the importance of observation in his quote and hence nicely captures the recurring theme of this thesis. Observation has taught us important lessons in most if not all areas of science. But also in everyday situations, we learn through observation. For example in a social interaction we may slowly adapt to a person’s behavior as we notice distress or anger based on behavioral queues. Or, to find our destination, we walk and observe our surroundings to figure out where we are and which way to go.

Information has become easier to access, as smartphones are being adopted by the general population. Also, these devices contain many sensors like microphones, cameras, GPS receivers, gyroscopes, accelerometers, and magnetic field sensors. Smartphones have gone from providing raw information to helping us observe our environment. For example, a smartphone can use

GPS signals to accurately track its position even if we put it into a, for us, unknown location. The devices' motion can even be tracked by inertial sensors without needing to receive any radio signals. These devices are becoming adept observers of the environment. However, even in well established research areas such as localization, there are still unresolved issues. For example, localization usually stops working when moving into an indoor environment. Mostly because GPS signals are heavily attenuated and hence reception is not possible.

Even if we may find it easy to navigate familiar indoor environments, localization would still be helpful to find lost items, to improve a shop by reorganizing items, or to efficiently route a robot vacuum. We will discuss how device motion and the surrounding WiFi infrastructure can be used to simplify and improve indoor localization in Chapter 2. Device motion and WiFi signal strength measurements can be combined to generate accurate signal strength maps. To some extent, people do the same thing when walking through a new environment. We remember where we are coming from but since this is inaccurate, we may loose track and find it hard to return to our place of departure. As soon as we recognize a place we have already visited, we can account for some of the confusion and improve our guess on where we are. We use the same principle to improve indoor localization. First, we show how device motion can be estimated through inertial sensors by modeling how people move while walking. This allows us to count steps and measure length as well as the changes in heading which we use to track a user's location. Second, instead of visual landmarks, we use the signal strength measurements to recognize previously visited locations. These landmarks help us to account for accumulating errors of the motion estimation which leads to diverging tracks. The resulting maps of WiFi signal strength landmarks are used to localize a device.

In history, people have developed advanced methods of localization that do not require knowledge of local landmarks, but still function on a global scale. Celestial navigation and compasses allowed us to find new continents and navigate environments that only present few landmarks such as the high seas. Observations such as the location of stars or the sun are not fixed for any given location. Hence, recognizing a location based on such an observation requires a more sophisticated method than comparing current to previous observations. If a measurement of inclination is incorrect, the location estimate may be completely wrong. An experienced navigator may be able to correct the error by using prior information and checking the observations for inconsistencies. However, constellations may be invisible during the day or under a cover of clouds. Again, a skilled navigator may be able to localize requiring only a fraction of stars to be visible.

Analogously, GPS localization works on a global scale as long as satellites are clearly visible. Also, localization can be disturbed if only one satellite's signal arrival time measurement is erroneous. We present an alternate approach to use satellite signal observations to deduce a receiver's position in Chapter 3. We focus on minimizing the effect of erroneous measurements. Instead of observing each satellite's signal arrival time individually, our receiver design assigns a likelihood value to a given location given the observed signal. Much like in the star example, as long as there is a sufficient number of visible satellites, our receiver delivers the most likely location without being distracted by erroneous measurements. This becomes increasingly important as the duration of the observed signal is shortened due to noise and non-line-of-sight effects. Our receiver can compute a position estimate based on a single millisecond of signal recording while maximizing the likelihood of successfully finding the correct position fix.

Digressing from the topic of localization, we show how social interactions can be tracked using smartphones. As voices originate from different locations, we show how to annotate speaker activity in audio to analyze the dynamics of a given discussion. In Chapter 6 we show how a single smartphone and its built in microphone allows to distinguish speakers in most situations. We extend this idea in Chapter 7 and discuss how a set of smartphones can be used to collaboratively annotate speaker activity in a distributed manner. The distances between the microphones and the participants cause varying propagation delays for different speakers which we exploit in our application. Relying on general purpose hardware, our solution handles clock inaccuracies and non deterministic system delays.

The introduction of smartwatches has brought even more inertial sensors into everyday life. Since these devices closely follow the motion of a user's wrist, these sensors can measure hand motion in great detail. We use this fact, to develop a gesture recognition system. To test the performance, we use motion data recorded from people writing on a whiteboard. This has the advantage that people do not have to memorize a large number of new gestures. Also most people are able to recreate their writing with high accuracy which should simplify the recognition process. Although the recognition rates of isolated letters is high, sequences of letters have proven to be hard to segment.

However useful the discussed observations may seem, they can also be exploited to infringe the privacy of users in unexpected ways. For example, a geotag in a holiday photo that is uploaded to a public website may be used to deduce a user's location at a certain time. In Chapter 4 we discuss how a skillful attacker can even infer touch input on a smartphone, solely based on how the phone moves during the input. The required motion data can be accessed by any application installed on current smartphones. Yet the touch input data is generally only available to the currently displayed application for security reasons. As the motion data is sensitive to the environment in which it is recorded, we can not convincingly show that inference is possible in a lab setting. Hence, we implement a smartphone game which collects motion and touch input data from many different users in completely uncontrolled situations. The results show that touch input inference is difficult but feasible. Especially for inputs that are repeated over and over, such as pins and passwords that unlock smartphones.

1.1 Collaborations and Contributions

This thesis is based on several publications and drafts I worked on during my time as a PhD student at the Distributed Computing group at ETH Zurich under the supervision of Prof. Dr. Roger Wattenhofer. Research often is a collaborative effort. I shall list in the following the publications or drafts underlying the respective chapters along with a list of the respective co-authors. Note that the authors of the papers are listed in alphabetical order and therefore do not represent the degree of contribution of the individual authors. Besides this, all articles listed below are joint work with my supervisor Prof. Dr. Roger Wattenhofer.

Chapter 2 is based on the publication *A Pocket Guide to Indoor Mapping* [17]. Co-author was Samuel Welten.

Chapter 3 is based on the publication *Fast and Robust GPS Fix Using One Millisecond of Data* [15]. Co-author was Manuel Eichelberger

Chapter 4 is based on the publication *Inferring Touch from Motion in Real World Data* [13]. Co-authors were Philipp Brandes and Jonas Passerini.

Chapter 5 is based on the publication *Recognizing Text Using Motion Data From a Smartwatch* [4]. Co-authors were Luca Ardüser and Philipp Brandes.

Chapter 6 is based on the publication *RTDS: Real-Time Discussion Statistics* [14]. Co-authors were Klaus-Tycho Förster and Jan Deriu.

Chapter 7 is based on the publication *Distributed Discussion Diarization* [16]. Co-authors were Klaus-Tycho Förster and Simon Tanner.

2

WiFi Indoor Localization

Today, GPS is an essential component of the global information infrastructure. GPS satellites surround our globe and current smartphones are equipped with suitable receivers. However, GPS based localization still has blind spots, in particular if a user enters a building. Often, navigating inside an unknown building can be just as hard (if not harder!) than navigating outside.

In this chapter, we exploit WiFi signal strength observations as well as the motion of a smartphone to deduce its location. We present a way to obtain accurate WiFi signal strength maps that can be used to localize any WiFi enabled device, without dedicated hardware, and without a time consuming and expensive training process.

The first contribution is a novel motion estimation technique that allows us to gather accurate user motion data in an unobtrusive way. The required sensors, a 3-axes accelerometer and a 3-axes gyroscope, can be found in many modern smartphones. Using these sensor measurements, we show how the direction of the leg and hip can be accurately estimated, independent of how the smartphone is placed in the trouser pocket. Based on these estimates, we can track the heading as well as the distance of each step the user takes. This motion model is tailored to fit the requirements of tracking a user's motion in indoor environments and therefore does not utilize the (often unreliable) magnetic field to determine the absolute heading. Instead, the change in heading between steps is estimated. We evaluate the performance of this motion model and show that, despite our non-restrictive sensor placement and the lack of absolute heading, the motion model is able to extrapolate from a single configuration parameter to work at different walking speeds for different people. The motion data is highly accurate for short tracking intervals, i.e. short walks with a smartphone do exhibit a small relative positioning error.

The positioning error is growing with distance though, so we use information about the signal strength of access points of the wireless infrastructure to globally correct that. Our second contribution lies in the integration of the complementary information – the “locally accurate” motion data and the “globally accurate” signal strength data – into a system that allows for easy mapping with nothing but a smartphone in the pocket. More precisely, we obtain additional positioning constraints between locations that the user has visited while walking around in the area of interest. We use the common (least-squares) Graph-SLAM technique to obtain corrected positions for all the signal strength measurements that a user recorded. As a result, the system is capable of recovering accurate relative positions of the signal strength measurements from data that was captured by multiple users walking through an area at different times. We show a qualitative example for the mapping performance in a large university building.

2.1 Related Work

Crowd-sourcing Localization. Recent work presented by Rai et al. [70] shows the high interest in localization solutions relying on crowd sourced data rather than time consuming training. Their system can provide localization information using only crowd-sourced information. Motion estimates and WLAN signal strength measurements are fused using a particle filter approach which also requires a floor plan map. As a result, the locations of the signal strength measurements within the floor plan can be estimated. High localization accuracy is achieved. However, the required floor plan map has to fulfill strict requirements in order for the particle filter to converge to the correct WLAN signal strength measurement locations. The probability density that has to be estimated with a particle has four dimensions which implies a high computational complexity on the mobile device. Our approach does not require a map and the use of efficient sparse non linear solvers that are readily available, the computational complexity is lower.

Signal Strength Mapping. The problem of acquiring signal strength maps in an unsupervised manner has been approached by Chintalapudi et al. [26]. The system they presented allows WLAN signal strength based localization without annotated training data. No motion data is used in the process. Instead, the relative positions of WLAN Access Points and mobile node locations are modeled as a function of the received signal strengths. The resulting set of overdetermined geometric constraints is solved using a genetic algorithm. Occasional GPS location measurements are used to obtain location information in the global frame. However, GPS measurements are hard to obtain in indoor environments and may be erroneous. The received signal strengths are the only indication for geometrical distance. Due to non-line of sight effects in indoor environments the complexity of this relationship is beyond the reach of the simple model used in the paper. This shortcoming becomes evident as the localization accuracy drops if the accurate locations of the access points are provided in the training phase. Therefore, the model might be accurate enough to provide localization capabilities, but the estimated map will in reality never converge towards the accurate access point distribution. In our approach, a person's motion is estimated in order to get geometrical distance information. The measured signal strengths are only utilized to recognized previously visited locations. Therefore, even if associations are incorrect, our maps converge to the accurate signal strength distribution because the increasing number of unbiased distance estimations from our motion model.

Motion Estimation. The dead-reckoning method presented in this chapter is the enabling factor that allows us to create signal strength maps. Many such methods to estimate human motion using foot-mounted sensors have been described in the literature [11, 35, 37, 63]. These solutions rely on the foot being stationary when in contact with the ground. This zero-velocity interval can be used to counteract the velocity estimation drift caused by integrating the accelerometer signals. An overview of different methods to estimate motion based on foot-mounted sensor measurements have been compared by Skog et al. [75]. Another approach, relying on sensors fixed on a helmet was presented by Beauregard [10]. As our goal is to perform mapping in an unobtrusive way, we desire to measure user motion based on its natural location when users are casually walking around. Therefore, all motion estimation methods that require dedicated hardware such as bodily fixed sensors are useless if we want to allow a wide range of users to contribute in the mapping process. Our motion estimation method only requires a smartphone being placed in a trouser pocket which makes it tangible for many people now, and even more so in the future. A motion estimation method that is seemingly similar to the one presented in this chapter was described by Blanke and Schiele [18]. This method and a selection of others that are equally nonrestrictive about the positioning of the sensors have been experimentally compared by Steinhoff and Schiele [76]. While the motion direction is estimated using the available sensor measurements, the step length was assumed to be constant and even manually set for each track. The presented evaluation does not allow us to quantitatively benchmark our approach because the evaluation results are distance- and orientation error quantities for each stride. The results are median stride length errors of more than 10cm and median orientation errors of more than 5° whereas we achieve mean localization errors of below 10% of the traveled distance. However, the argument that an estimation error of 10cm per stride is already more than 10% of the traveled distance is highly questionable. Partly due to measurement errors in the ground truth but also because our approach estimates not only step- heading and count but also distance. Also our motion model can cope with different walking speeds without requiring the stride length to be manually configured. In addition to this, our approach does not rely on the use of magnetic field sensors to determine the absolute heading direction. Due to the large magnetic disturbances that may interfere with finding the correct direction towards north, using magnetometers in indoor environments is questionable and may incur large and unexpected errors. As a result, our approach is not able to estimate absolute headings for the steps, but only the change in heading for consecutive steps. Another system that is seemingly similar to ours, but requiring two separate, fixed sensors to estimate motion based on the orientation of the thigh was presented by Lee and Mase [46].

SLAM. In the robotics community, the Simultaneous Localization and Mapping Problem (SLAM) has been an active research topic for over twenty years. Bailey and Durrant-Whyte [8, 31] summarized the most important results and solution ideas. To apply these SLAM solutions, two complementary information sources are required. Firstly, the motion of the measurement device has to be estimated. Secondly, previous locations have to be recognizable if visited again. Both requirements can be met with off-the-shelf smartphones. The known SLAM solutions already have been applied to the indoor mapping problem in several instances. Ferris et al. [34] introduced a method to build signal strength maps in indoor environments based on Gaussian Process Latent Variable models. Motion information is integrated with signal strength observations to estimate the signal strength distribution which can be used for localization. While the approach is able to reconstruct topologically correct maps, the true geometric shape of the building could not be captured. Comparing their reconstructed maps to ours (see Figure 2.6) clearly indicates the superior mapping accuracy of our approach. Huang et al. [40] presented an adaption of the Graph SLAM method which uses WLAN signal strengths as observations. Pedometer and gyroscope measurements are used to estimate the user's motion. The results show that signal strength measurements are sufficient to recognize previously visited locations. We follow this approach and formulate a sparse non-linear optimization problem based on the motion estimates and signal strength measurements. This has the advantage that many highly efficient solvers are available and can directly be applied to solve the mapping problem. Also this problem formulation allows us to combine motion and observation information from multiple walks or even users. In the publication of Huang et al. [40], the resulting pose distribution is only shown for a small building that allows for many loop closure constraints. Also, it is unclear how to crowd source the required data as the motion estimation approach is assumed to be known. Our motion estimation approach allows us to present results for larger buildings with sparser loop closure opportunities. Additionally, our approach requires only a smartphone as well as existing wireless infrastructure.

2.2 Motion

The algorithm we present to track the motion of a user requires a 3-axes gyroscope and a 3-axes accelerometer to be placed loosely in a trouser pocket. Modern smartphones not only contain the required set of sensors, but are likely to be located in a user's trouser pocket. Additionally, the proximity- and light-sensors can be used to determine if the phone might be located in- or outside the pocket. The method is split into three parts which are described in the three following subsections. Firstly we show how the orientation changes of the phone can be estimated using the gyroscopes only. These orientation estimates are then used to track the motion of the users thigh which allows us to find the exact moments in time when the leg reaches its extreme orientations as depicted in Figure 2.1. We then estimate the length and the direction of each step based on two consecutive orientation extrema.

Orientation. In indoor environments, the earth magnetic field is heavily disturbed by power cables, concrete reinforcements and such. In addition, magnetic disturbances are caused by varying electrical currents in the sensor frame itself. To complicate the use of these magnetic field sensors even more, infrequent recalibration of the sensors occurred without notification. Since these errors are hard to model, we decided to not use the magnetic field sensor to counter the heading drift caused by the gyroscopes. We found that the heading drift caused by the gyroscopes is nearly time independent and can be effectively corrected with the observation model presented later on. Therefore, we only use the gyroscope measurements $(\omega_x, \omega_y, \omega_z)$ to estimate

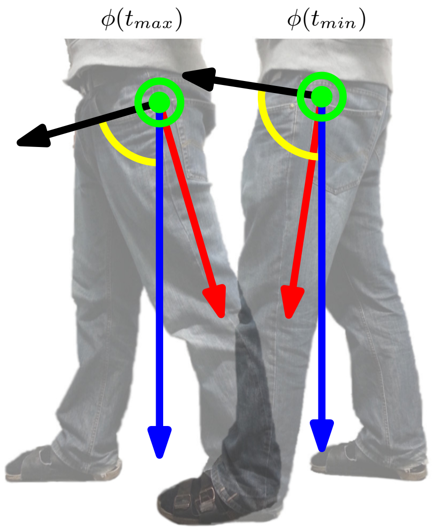


Figure 2.1: The most important vectors used to find $\phi(t)$ are shown for a minimum of $\phi(t)$ and a maximum of $\phi(t)$ respectively. The leg direction $\mathbf{L}$ is shown in red, the hip direction $\mathbf{H}$ is shown in green (points out of the image), the earth gravitational force $\mathbf{g}$ is shown in blue, the inclination angle $\phi(t)$ is shown in yellow and the vector used to find the inclination $\mathbf{L} \times \mathbf{H}$ is drawn in black.

the orientation quaternion $\mathbf{q}$ which can be computed using the Hamilton product:

$$\mathbf{q_t} = \mathbf{q_{t-1}} \cdot (1, \frac{\omega_x}{2}, \frac{\omega_y}{2}, \frac{\omega_z}{2}) \qquad (2.1)$$

This orientation estimate relates the sensor frame to an arbitrarily chosen coordinate frame which has an offset to the earth coordinate frame that evolves according to the gyroscope drift. While the orientation estimate drift leads to errors in the motion estimates, knowledge of the absolute orientation in the earth coordinate frame is not required.

Rotation Based Step Detection. The following step detection and estimation is based on the assumption that a smartphone that is placed in a trouser pocket approximately follows the motion of the thigh. We use this assumption to infer the evolution of the orientation of the thigh and the axis around which it rotates (the hip axis). We then use these vectors to find orientation extrema which we use to determine step direction and distance.

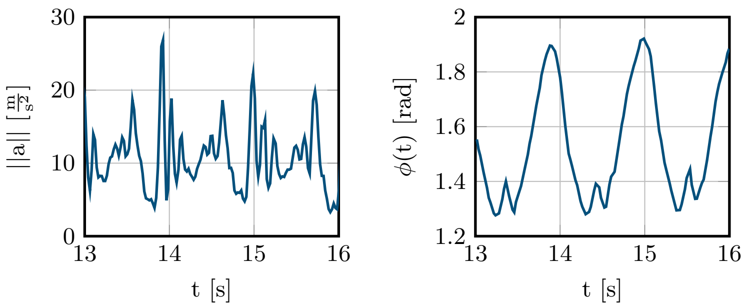


Figure 2.2: A comparison of the accelerometer magnitude (left) and inclination angle ϕ (right) during several steps on a treadmill. The treadmill setting for the measurements was set to $1\frac{km}{h}$. Clearly the inclination ϕ is suitable to detect steps.

The most relevant vectors that are used in the following discussion are shown in Figure 2.1. In the following discussion, all vectors are expressed in sensor coordinates. The acceleration of the earth's gravitational field $\mathbf{g}$ is transformed into the sensor coordinate frame using the orientation quaternion $\mathbf{q}$ from Equation 2.1.

In a first step, the direction of the leg $\mathbf{L}$ in the sensor coordinate frame can be estimated using a low-pass filter:

$$\mathbf{L} = L_C \cdot \mathbf{L} + (1 - L_C) \cdot \mathbf{g}$$

The cutoff frequency $L_C = \sqrt{\mathbf{L} \cdot \mathbf{g}}$ is chosen to allow the leg estimate to quickly converge whenever the device orientation relative to the thigh changes. During normal use, $\mathbf{L}$ is oscillating around $\mathbf{g}$ and the leg estimate is not significantly converging towards $\mathbf{g}$. The normalized estimate $\frac{\mathbf{L}}{|\mathbf{L}|}$ can be used to determine the rotation axis between the leg and the direction of the gravitational force:

$$\mathbf{r} = \mathbf{g} \times \mathbf{L}$$

Based on the rotation axis $\mathbf{r}$, we estimate the direction of the hip $\mathbf{H}$ using a low-pass filter:

$$\mathbf{H} = H_C \cdot \mathbf{H} + (1 - H_C) \cdot \mathbf{r} \tag{2.2}$$

The cutoff frequency $H_C = |sin(|\mathbf{r}|)|$ is chosen to increase the influence of the rotation axis $\mathbf{r}$ on the hip estimate $\mathbf{H}$ if the angle between $\mathbf{L}$ and $\mathbf{g}$ grows. The reason for this is the fact, that the rotation axis can be determined with higher accuracy if the angle between the two vectors is larger. In case the rotation axis is pointing into the opposite direction of the hip estimate, the rotation axis is reversed ($\mathbf{r} = \mathbf{r} \cdot \text{signum}(\mathbf{r} \cdot \mathbf{H})$) before applying the low-pass filter in Equation 2.2. The hip axis **H** converges to one of the two (opposing) main rotation axes of the leg. In the absence of absolute heading information, the two convergence possibilities of **H** deliver equal estimation performance and results. The hip- and leg-direction estimates are then used to find $\phi(t)$:

$$\phi(t) = \arccos\left((\mathbf{L} \times \mathbf{H}) \cdot \mathbf{g}\right) \tag{2.3}$$

The evolution of $\phi(t)$ is used to find the points in time when the the thigh is at the extrema of the back and forth movement. Figure 2.2 visualizes $\phi(t)$ and for comparison also shows the evolution of the acceleration magnitude signal $a(t)$. In $phi(t)$ the minima and maxima are easy to detect whereas in the accelerometer signal, it is extremely hard to extract isolated steps. Each minimum and maximum in $\phi(t)$ corresponds to the end of the last step and the beginning of the next. With accelerometer based step detection methods, counting steps gets increasingly difficult at low walking speeds, as the foot impact gets less articulated in the accelerometer signal. In addition to this, $\phi(t)$ allows us not only to count steps, but also to find the orientation extrema of the leg swing which allows us to accurately estimate the step length and direction.

Step Estimation. The direction and length of each step is estimated based on the smartphone orientations recorded in the minima and maxima of $\phi(t)$. The change in orientation between minima and maxima can be expressed as axis **a** and angle α. Because absolute heading information is not available, **a** can be used as the walking-direction. The angle α can be used to estimate the step length.

$$\mathbf{s} = \frac{\mathbf{a}}{|\mathbf{a}|} \cdot c \cdot \sin\left(\frac{\alpha}{2}\right) \tag{2.4}$$

The constant c is user dependent and has to be configured.

2.3 Observation

In addition to the motion data, the smartphone is able to record the evolution of received signal strength indicators (RSSI) for all visible access points. In indoor environments, inferring distance from received signal strengths is infeasible. Non-line of sight effects and antenna imperfections lead to a spatial signal strength distribution which cannot be captured in a function that depends on the distance between sender and receiver. This is why we use these signal strength measurements not to infer physical distance, but to recognize previously visited locations only. We achieve this recognition by comparing two landmarks L_1 and L_2 using the following signal space

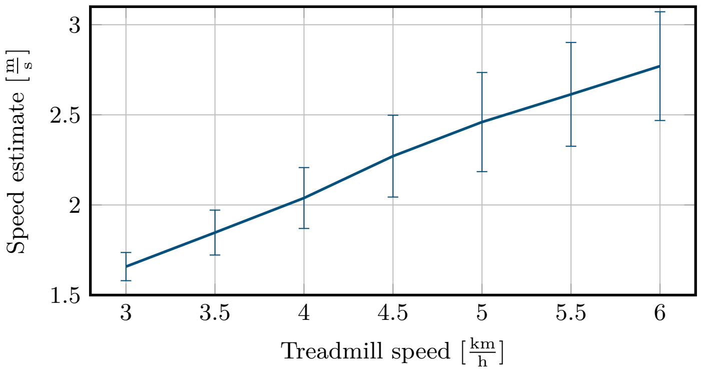


Figure 2.3: Comparing the motion model speed output to the ground truth (Treadmill speed setting). Clearly, the motion model output linearly depends on the actual walking speed which means that when properly configured, the motion model will work at different walking speeds.

distance measure:

$$\begin{aligned} s(L_1, L_2) =& \frac{1}{|M_1 \cap M_2|} \sum_{e=M_1 \cap M_2} |M_1(e) - M_2(e))| \\ &+ \frac{1}{|M_1 \setminus M_2|} \sum_{e=M_1 \setminus M_2} |M_1(e) - l_{min})| \\ &+ \frac{1}{|M_2 \setminus M_1|} \sum_{e=M_2 \setminus M_1} |l_{min} - M_2(e))|. \end{aligned} \tag{2.5}$$

The sets of visible access points are denoted as M_1 and M_2 respectively. Access points that report an RSSI lower than l_{min} are neglected. In addition, access points that are only visible in the other landmarks are considered to be received with signal strength l_{min}. Two landmarks assumed to be captured in the same location (associated) if their distance measure $s(L_1, L_2)$ is lower than a threshold s_{th}.

2.4 Fusion

The complementary characteristics of the motion- and observation association constraints can be exploited to counteract the divergence of integrating motion estimates using the observation associations. Similar to finding the equilibrium point of a system of springs and masses, the visited locations correspond to the masses and are described as point in space as well as current motion heading $x_i = (x, y, \phi)$. The motion and association constraints correspond to the springs. The stiffness of the springs corresponds to the confidence level of the estimates. Whereas the association constraints are modeled as springs with equilibrium length zero and do not constrain the heading difference, the motion constraints are modeled as springs with equilibrium length equal to the estimated step length that also constrain the change in heading ϕ between consecutive poses.

$$x_i = f_i(x_{i-1}, u_i) + w_i. \tag{2.6}$$

The pose x_i is linked to the previous pose x_{i-1}, using the sensor readings u_i. The model uncertainty is captured in the Gaussian noise term w_i. Landmark associations are described as follows:

$$0 = ||x_{i_k} - x_{j_k}|| + v_k. \tag{2.7}$$

This captures the fact that if we associate two landmarks, we expect them to be recorded in spatially close locations. The Gaussian noise term v_k may vary depending on the association quality where k addresses one specific association between two poses x_{i_k} and x_{j_k}. The optimization problem is defined as follows:

$$\begin{aligned}\Theta^* = \underset{\Theta}{\operatorname{argmin}} \Bigg[& \sum_{i=1}^{M} ||f_i(x_{i-1}, u_i) - x_i||^2_{\Lambda_i} \\ & + \sum_{k=1}^{K} ||x_{i_k} - x_{j_k}||^2_{\Sigma_k} \Bigg].\end{aligned} \tag{2.8}$$

For simplicity, the notation $||e||_\Sigma = e^\intercal \Sigma^{-1} e$ was used. The solution is the set of poses Θ that minimizes the given cost function based on the constraints obtained from the motion- and observation model. We solve the sparse non-linear optimization problem using iSAM [42].

Compared to a particle filter based approach, this non-linear least squares problem allows us to overcome the lack of absolute heading information as well as heading drift without increasing the computational complexity. Note that a particle filter would need to sample the joint probability function of spatial location, heading offset and heading drift. Adding dimensions to the probability distribution causes the number of particles to grow exponentially.

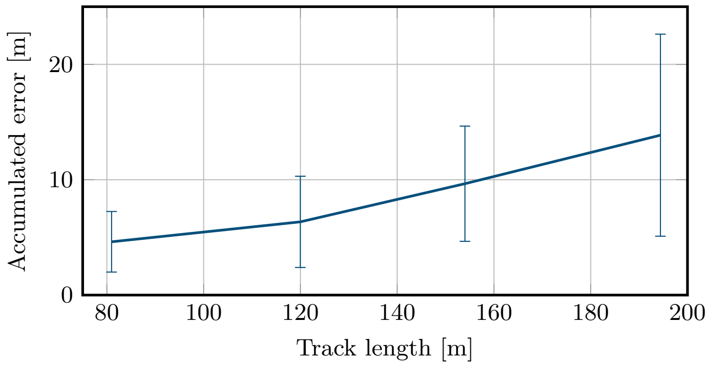


Figure 2.4: Relating the traveled distance with the resulting motion model integration error.

2.5 Localization

As a sample Application, we propose a localization scheme based on the acquired signal strength fingerprints. Localizing a signal strength measurement is achieved using the k-nearest neighbors in the signal strength space:

$$\mathbf{p} = \frac{1}{\sum_{i=1}^{K} d_i} \sum_{i=1}^{K} d_i \cdot \mathbf{p_i}. \tag{2.9}$$

Here, d_i are the signal space distances defined in Equation 2.5, $\mathbf{p_i}$ are the corresponding landmark locations of the k-nearest neighbors and $\mathbf{p}$ is the localization estimate.

2.6 Experiments and Results

The following experiments were carried out using a Samsung Nexus S smartphone. In a first step, the motion- and observation models are evaluated separately because the motion model will deliver user specific performance whereas the observation model will be largely user independent.

Motion. Firstly the speed estimation is compared to the actual walking speed. A single user was walking on a treadmill whose speed setting was used as the ground truth walking speed.

The results shown in Figure 2.3 indicate a linear dependance between actual speed and motion model estimate. This is the desirable outcome

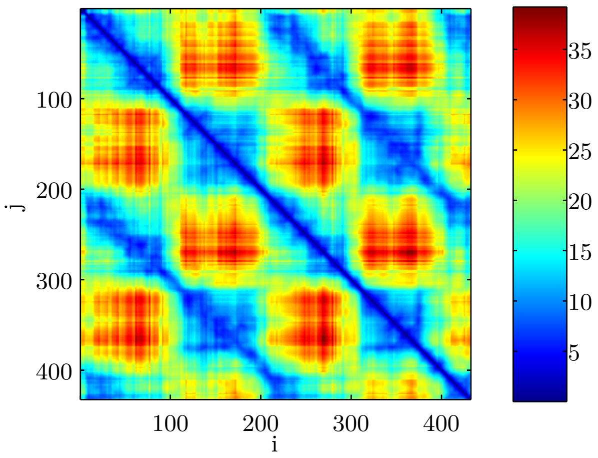


Figure 2.5: While walking around a set of rectangular shaped (50m x 10m) set of hallways twice, roughly 420 fingerprints were collected. This figure shows the fingerprint signal space distance for all pairs i, j of recorded fingerprints.

since it indicates that the motion model can be configured for a specific user with only one parameter and deliver accurate speed estimates over a variety of walking speeds. The variances shown for each treadmill speed setting indicates a large variance of of the speed estimates even though the treadmill- and therefore walking speed was very constant during the experiment. The results shown in Figure 2.3 were used to find the parameter c from Equation 2.4.

The distance estimation was evaluated with eleven people that were asked to walk down a 51 meter long straight hallway. The motion model output had a mean of 51.7 meters and a standard deviation of 4.4 meters. This result indicates, that the motion model may be adapted for different users. To evaluate the motion model accuracy in a more realistic setting in which people were walking through hallways, doors and bends, the same eleven people were asked to walk along four predefined tracks with increasing lengths (51m, 81m, 120m, 154m, 195m). All the tracks ended in their starting point which means, the localization error after each track is captured in the distance between start- and estimated end-point. Figure 2.4 shows the increasing localization error caused by accumulating motion estimation errors. As expected, the localization error increases as the traveled distance grows. A large portion of the error is caused by the drifting heading estimate which could not be corrected using the magnetic field sensors. Although the closed tracks facilitate evaluating the localization error, note that this evaluation scheme does not capture the fact that the motion model might systematically over- or underestimate the step lengths for different users. However, combined with the results shown in Figure 2.3 and the distance measurement accuracy of the straight hallway experiment we conclude that the motion model works for a variety of people walking at different speeds only requiring to be configured at one speed, or even only using the users body height.

Observation.

Observations are associated to each other if their mutual signal space distance is below a given threshold. Therefore, we require the signal space distance to be low for two spatially close fingerprints. On the other hand, comparing spatially distant fingerprints should lead to a high signal space distance. To evaluate how well this requirement is met by the smartphone WLAN observations, we carried out the following experiment. We collected a large number of fingerprints by walking at constant speed through office building hallways. The hallways are are forming a rectangle (50m x 10m) which is traversed twice during the experiment. The distance measure for all the pairs of recorded fingerprints are shown in Figure 2.5. Note that the distance measure is commutative and therefore, the distribution is symmetric. Also, elements that are close to the diagonal indicate signal space distances of two fingerprints that have been recorded within a short period of time. The farther away from the diagonal, the larger the time span between the two recorded fingerprints. Because the experiment was conducted while walking at constant speed, this also means that the time span in which two fingerprints were recorded linearly translates to a spatial distance. Since the goal of this experiment is to get an idea of how well our signal space distance measure is able to distinguish fingerprints that are far from fingerprints that are close, it is desirable to have low distance measures along the diagonal, but high distance measures the further away from the diagonal. In addition to the main diagonal, two secondary diagonals which are originate from the second round through the rectangular set of hallways. Similar to the main diagonal, we desire low distance measures close to the secondary diagonals but high distance measures the further away we get. Clearly the secondary diagonals are less clear than the main diagonal especially, the off diagonal elements are less distinguishable from the diagonal elements. The four by four checkerboard pattern is a result of the rectangular track and shows that fingerprints recorded in the two opposite long hallways have a large signal space distance. However, within one hallway, the spatial distance of fingerprints with small signal space distances can grow large. Due to this shortcoming, erroneous associations have to be expected. In addition, the

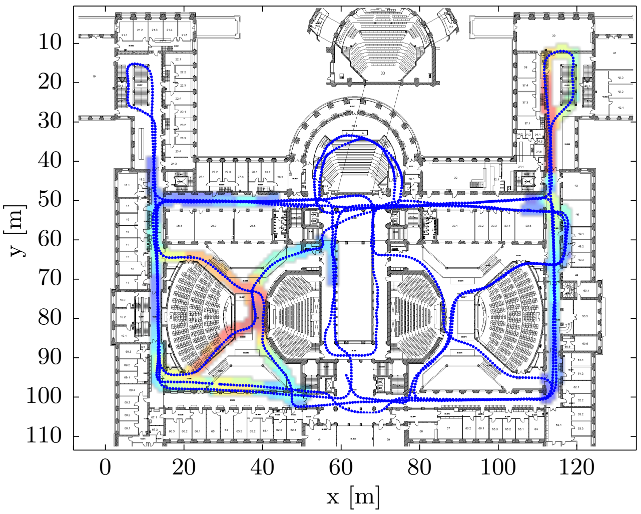


Figure 2.6: Fingerprint distribution for ETH Zurich main building (HG). Fingerprint locations are indicated as blue dots. Signal strengths for selected access points are shown in blue (weak) to red (strong) colors

similarity between fingerprints within one hallway will impede localization performance.

Fusion. Fusing motion- and association data by non-linear optimization leads to a map of fingerprints. The quantitative evaluation of the map quality is difficult as long as no localization scheme is used to measure localization performance based on the generated fingerprint map. Figure 2.6 shows the fingerprint distribution resulting from the fusion step for the ETH main building. The recorded fingerprints are drawn as blue dots. The received signal strengths for two distinct (non overlapping) access points are indicated in blue to red colored areas around the corresponding fingerprints. The relative fingerprint positions are the result of the fusion step. The floor plan as well as the rotation and translation between floor plan and fingerprint distribution is not automatically obtained but added by hand. The fingerprint locations obtained in the fusion step resemble the true shape of the building with with few exceptions (intersection on the right side).

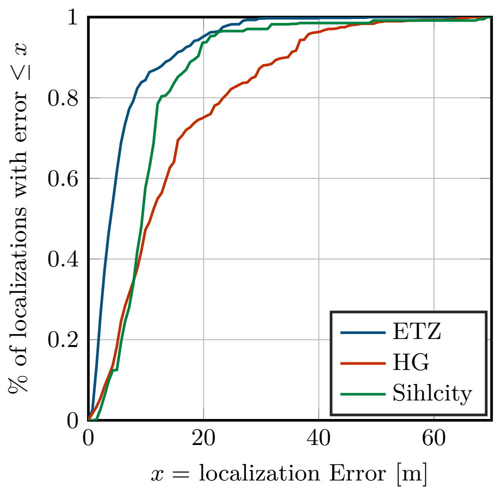


Figure 2.7: Localization performance in the three different buildings using KNN localization scheme described in Equation 2.9

Localization. The fingerprint distributions for all three experiments were comparably accurate as the one shown in Figures 2.6. One explanation for the poor performance is that buildings consisting of large open spaces lead to more dispersed signal strength distributions than buildings that contain only small open spaces. This means that smaller measurement differences translate to larger localization differences and therefore, the system is more sensitive to measurement errors. The performance difference mimics the environments in which the experiments were performed. The sites range from an office building (ETZ), a university main building with large hallways and lecture halls (HG), to a shopping mall (Sihlcity).

2.7 Conclusion

In this chapter we discussed how smartphones that are carried in trouser pockets can learn signal strength distributions. To this end, a method to track how people walk was introduced. Locations can be identified from the received signal strengths of visible WiFi access points. Hence, returning to a known location allows the system to compensate for errors that accumulate slowly when tracking people's movements.

3 Fast and Robust GPS Localization

In the previous chapter, we discussed localization in indoor environments which provide little to no GPS reception. In this chapter, we show how the existing GPS infrastructure can be used in a novel receiver design that improves performance. For most outdoor scenarios, GPS is the localization system of choice, mainly due to its global coverage and accuracy. However, continuous receiver operation consumes too much energy for mobile devices such as fitness trackers or even smartphones. Mostly because current receivers cannot be efficiently duty-cycled. When the receiver is switched off for a few minutes to conserve power, it takes a lot of time and energy to compute a new position fix once it is turned back on again. This has far-reaching consequences for many application scenarios. For example, today's GPS receivers make us wait for a first fix, which can be annoying if one wants to navigate an unknown place. Also, geotagging photos is not instant and energy consuming. Due to the energy consumption issues, many applications, such as long term tracking, are out of reach.

In this chapter, we present a receiver which requires only a single millisecond of GPS signal to compute its position. This means that the signal can be recorded and stored locally for later processing. Alternatively, the signal recording can be sent to a remote server which can perform the energy consuming position computation. This translates to a reduction in energy consumption as well as an increase in convenience for many applications.

For example, the initial position when navigating with your phone can be found within a few milliseconds depending on network latency. A smartwatch or fitness tracker may be able to track its location every few seconds for weeks at a time. When the duty cycle is further reduced, a tracking device that only requires one position fix per hour may run for years on a single coin cell battery. Geo-tagging photos can be simplified to adding a one ms signal recording to the photo which is stored and from which the position can be computed later on.

To increase the noise tolerance as well as the robustness in non-line of sight situations, we search the most likely position at which the recorded signal was received. This reduces the impact of satellite range estimation errors which can throw off current GPS receivers as well as CTN receivers based on the classical least squares localization approach. Since we rely on a single millisecond of signal, noise tolerance is even more critical compared to receivers running tracking loops and long integration intervals.

If the location is approximately known, finding the most likely position can be achieved by computing the likelihoods of all close-by positions and selecting the most likely one. The more uncertain the initial guess about position and time, the larger the search space (position and time) gets. Computing all the likelihoods presents a computationally expensive maximization problem in this case. However, we show how the global maximum can be found efficiently using a branch and bound approach. The runtime of the algorithm is correlated with signal quality: In good signal conditions, the computational load is low. The worse the signal conditions get, the higher the computational burden. However, the most likely position and time fix is found in any case. The branch and bound implementation speeds up the acquisition time and hence also the *time to first fix (TTFF)*.

We exploit the shape of the likelihood function to achieve higher positioning accuracy and robustness. As a result, under similar conditions (signal duration and sampling rate), our method leads to more accurate positioning compared to previous approaches. Furthermore, we show that that there is a trade-off between the amount of sampled signal used and the accuracy of the positioning solution. If we average over two consecutive fixes from one millisecond of data each, the median error is reduced from 25 to 15 meters. Averaging over 30 fixes (0.03 s of signal), the median error is as low as 6 meters. Tracking a user's position decreases the computational complexity of each consecutive fix as the search space (space and time) is much smaller.

3.1 Related Work

Van Diggelen [81] introduced the idea of *Coarse Time Navigation (CTN)*. Using CTN, a position fix can be found from only a few milliseconds of data without decoding any data from the GPS signal. The requirement for this is prior knowledge of the receiver time and position to within a few seconds and 150 kilometers, respectively. Liu et al. [49] showed that since CTN only requires a few milliseconds of data, the raw signal can be stored and the computation can be outsourced or postponed until power is available. This mitigates the problem of high energy consumption for acquisition by not acquiring the satellites on the receiver, enabling duty cycling. However, due to the short signal duration, accuracy and robustness is worse than in classic receiver designs relying on acquisition and tracking stages. Our GPS receiver design extends this idea and can compute a position from a *single* millisecond of signal. Our localization method counteracts the effect of the short signal duration and improves positioning accuracy compared to existing work on CTN. Also, we show how accurate position fixes can be computed from inaccurate time estimates. This allows us to drop the heavy and power consuming DCF-77 clock receiver required by Liu et al. [49]. As a result, our receiver can be miniaturized and function for years even when there is no clock synchronization except at the very beginning.

A second branch of research is concerned with improving the robustness of GPS receivers. In classical GPS, the receiver location is determined based on signal parameters. The most important ones being Doppler shift and code delay for each satellite. From these parameters, a position in space is computed. Clearly, signal parameters may be erroneously detected which leads to erroneous position estimates.

Instead of estimating the signal parameters, Closas et al. [29] showed how the receiver position can be estimated directly and how this can improve the robustness of GPS receivers.

We refer to the basic idea as *Collective Detection (CD)*, but it is also called *Direct Positioning* or *Combined Detection* in the literature. We improve the robustness of our approach by applying CD and we show how the computational complexity of CD can be reduced without giving up any accuracy.

Evaluations of CD have been performed in both simulation and practice [6, 25, 29]. The main concern is the computational complexity introduced by the high-dimensional search space. Also, the likelihood function is generally not smooth, such that standard greedy maximization methods cannot always find the global maximum. Optimizations such as the one proposed by Axelrad et al. [6] reduce the computational complexity but cannot guarantee that the best possible position is found. Our branch and bound implementation reduces the computational complexity and always yields the globally best solution.

3.2 GPS Fundamentals

The method described in the following is not only applicable to GPS, but also to other global navigation satellite systems (GNSS) such as Galileo[1] or GLONASS[2] with minor adaptations.

The GPS system can be seen as having three parts: the control segment, the space segment and the user segment. The space segment consists of 24 satellites orbiting the Earth [30]. A network of monitor stations and ground antennas makes up the control segment. It is primarily used to monitor the satellites state and keep track of their ever changing orbits. The orbits need to be known as accurately as possible to improve localization performance [30][3]The third – and for our discussion most important – part of GPS are the receivers, making up the user segment.

3.2.1 GPS Signal

The satellites transmit signals in different frequency bands. These include at least the so-called L1 and L2 frequency bands at 1.57542 GHz and 1.2276 GHz [30]. The signals are transmitted through a helix array antenna which right-hand circularly polarizes the signals [30]. This helps suppressing multipath signals at a receiver because a reflection of the signal polarizes it in the opposite direction.
In order to distinguish the signals from different satellites and to extract the signals from the background noise, code division multiple access (CDMA) is used.

Figure 3.1 shows the modulation scheme utilized in GPS. The Coarse/Acquisition code (C/A code) is a sequence of 1023 bits which is satellite specific. Gold codes are used to achieve favorable correlation and cross-correlation properties [81]. The C/A code is transmitted at 10.23 MHz which means it repeats every millisecond. The data is transmitted at 50 $\frac{\text{bit}}{\text{s}}$ and hence, each bit contains 20 complete C/A cycles. The data and C/A code are merged using an XOR before being mixed with the L1 or L2 carrier. Figure 3.1 shows how the GPS signal is generated. Note that the C/A frequency and the L1 frequency do not have the correct ratio to make the figure easier to read. The data that is broadcast contains a timestamp (called HOW) which can be used to compute the location of the satellite when the packet

[1]`http://galileognss.eu/`

[2]`https://www.glonass-iac.ru/en/`

[3]Further information can be found at `http://www.gps.gov/systems/gps/control/`

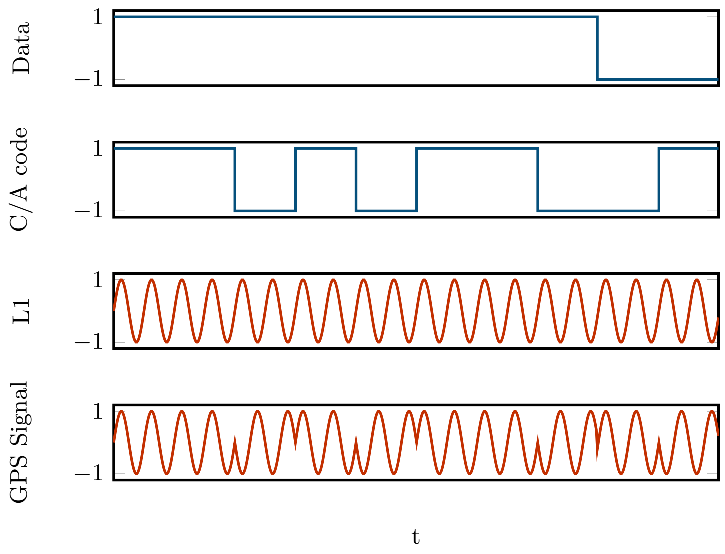


Figure 3.1: The structure and modulation of the GPS Signal. The binary data and C/A code are mixed with the carrier frequency (L1) using the BPSK modulation scheme.

was transmitted. However, to do this the receiver needs accurate orbital information (called ephemeris) about the satellite which changes over time. While the HOW timestamp is broadcast every six seconds, the ephemeris data can only be received if the receiver can decode at least 30 seconds of signal.

3.2.2 Localization

Conventional GPS receivers use three stages when obtaining a location fix.

Acquisition. First, the set of available satellites has to be found. This can be achieved by correlating the received signal with the known C/A codes from the satellites. Since the satellites move at considerable speeds, the signal frequency is affected by a Doppler shift. Hence, receivers usually correlate the received signal with C/A codes with different Doppler shifts.

Tracking. After a set of satellites has been acquired, the data contained in the broadcast signal is decoded. Doppler shifts and C/A code phase are tracked using tracking loops. After the receiver obtained the ephemeris data and HOW timestamps from at least four satellites, it can start to compute its location.

Localization. Localization in GPS is achieved by multilateration, a technique using *time difference of arrival (TDoA)* measurements to compute a position. The arrival times of the HOW timestamps received in the tracking phase are used to compute the set of TDOAs.

In *trilateration*, the distances between a mobile station and some stations with known position are measured. This can for instance be done through *time of arrival (TOA)* measurements, when the signal transmission times are known and all the stations are synchronized in time. The position of the mobile station lies at the intersection of the spheres around the stations with fixed position with the corresponding radii.

While the satellites operate on an atomic frequency standard, the receivers are not synchronized to the GPS time. Therefore, the local time at a receiver is unknown. Due to that, the distance of the receiver to the satellites cannot be directly computed from the local arrival time of the signals at the receiver. Instead, only the time *differences* of the arrival times can be measured. Therefore, the multilateration method is applied. From a computational view, there is not a large difference between the trilateration and the multilateration approach. The latter problem contains one more variable, which is the receiver time, and hence needs measurements from at least four instead of three satellites for the problem to be well-defined. The receiver position is found through a least squares optimization.

3.2.3 Assisted GPS

One of the main disadvantages of GPS is the low bit rate of the navigation data encoded in the signals transmitted by the satellites. The minimal data necessary to compute a position fix, which includes the ephemerides of the respective satellites repeats only every 30 seconds. In order to decode all that data, the receiver has to continuously track and process the satellite signals which induces high energy consumption. Furthermore, upon starting up a receiver, a position will not be instantly available. To overcome this drawback, receivers can run continuously, but this consumes even more power.
Assisted GPS (A-GPS) drastically reduces the startup time by fetching the navigation data over the Internet, commonly by connecting via cellular networks which are faster than decoding the GPS signals and normally only takes a few seconds. Using this data, the acquisition time can be reduced since the set of available satellites can be estimated along with their expected Doppler shift. This is possible because the exact arrival time of the

navigation data is not required for the localization. Also, ephemeris data is valid for at least 30 minutes. However, the receiver still needs to extract the HOW timestamps from the signal but since they are transmitted every six seconds, this is roughly how much time it takes an A-GPS receiver to compute a position fix.

3.2.4 Coarse-Time Navigation

Coarse-Time Navigation (CTN) is an A-GPS technique which drops the requirement to decode the HOW timestamps from the GPS signals. Van Diggelen [81] describes the concept in detail. The only information used from the GPS signals are the phases of the C/A code sequences which are detected using a matched filter. These arrival times are directly related to the sub-millisecond parts of the corresponding TDoAs. The number of whole milliseconds of the signal propagation time are resolved with a known approximate location and time. Because the signals travel at the speed of light, which is about 300 km per millisecond, in order to be able to resolve the number of whole milliseconds unambiguously, the deviation may at most be 150 km from the correct values. Here, the deviation is defined as the local clock offset multiplied by the relative speed between the satellites and the receiver plus the initial position error. Since the PRN sequences repeat every millisecond, without considering navigation data bit flips in the signal, CTN can in theory compute a position from one millisecond of the sampled signal. But since bit flips can occur, to make sure all visible satellites can be used, two milliseconds are necessary. With such short signal recordings, clearly noise becomes a major issue, because noise cannot be filtered out as easily as with much longer recordings of several seconds. But the advantage of this extremely short recording period is that the signal can be recorded quickly and without consuming much energy. As a result, the TTFF can be shortened to less than a second. Also, since no metadata has to be extracted from the GPS signal, CTN may be able to compute a location even if the GPS signal cannot be decoded due to noise or attenuation.

3.2.5 Collective Detection

Collective detection builds upon the observation that detecting peaks in the correlation functions of individual satellites might yield sets of pseudoranges which are not consistent with the laws of physics. By searching a solution in space and time directly, this can be avoided. The problem then consists of finding the most likely position given the received signal. From a given hypothetical position and time (referred to as hypothesis in the following), the corresponding ranges of the satellites can be inferred, which are then used to determine the arrival times of the signals. A receiver can exploit this by combining corresponding correlation values from all the satellites to compute a likelihood measure. This is essentially what our receiver does. Erroneous peaks in the correlation function most likely do not align which improves noise resistance. Commonly, the hypothesis pseudo-likelihood is defined as the sum of the satellite pseudo-likelihoods, but one could also use other measures, for instance the product.

3.3 Localization Method

The basic idea of our method is to asses the quality of many hypothetical receiver states $h = (h^p, h^t)$ which consist of the receiver position h^p and time h^t. The quality of a hypothesis is determined through a function which assigns a pseudo-likelihood to the hypothesis given external information and the observed signal. This likelihood $\mathcal{L}(h)$ is a measure of how well the observed signal matches the signal expected at a hypothesis h.

3.3.1 Likelihood

Given a hypothesis h and the current orbits of the satellites, we can infer the satellites' signal transmit times and propagation delays to compute the expected signal phase $\phi_i(h)$ arriving at the receiver from the i^{th} satellite. This is discussed in detail in Section 3.3.2. Hence, for any hypothesis h we can expect a C/A code with phase $\phi_i(h)$ from satellite i in the arriving signal. We can evaluate how well the received signal $r(t)$ matches this expectation by computing a single correlation value with satellite i's C/A code $\mathrm{ca}_i(t)$.

$$c_i(h) = \sum_{\tau=0}^{1\text{ms}} |r(\tau) \cdot \mathrm{ca}_i(\tau - \phi_i(h))| \tag{3.1}$$

If our hypothesis h is correct, we expect large correlation values c_i for satellites whose signal can be received, because the code phase of the C/A code in the received signal match the expected code phase $\phi_i(h)$. For satellites that are heavily attenuated or reflected, c_i will be almost completely random. We define our likelihood function as the sum of the correlation values for a given hypothesis over all *visible* satellites, whose indices are denoted by the set V.

$$\mathcal{L}(h) = \sum_{i \in V} c_i(h) \tag{3.2}$$

The receiver position and time are estimated by selecting the hypothesis h^* which maximizes the likelihood measure:

$$h^* = \arg\max_{h \in F} \mathcal{L}(h)$$

where F is a set of *feasible* hypotheses (position, time tuples).

3.3.2 Computing the C/A Code Phase

To compute the likelihood of a hypothesis h, we need to know the C/A code phases $\phi_i(h)$ of the visible satellites. In the following, we assume that the signal propagation delay $d_i(h)$ is mainly determined by the distance between receiver and satellite. Note that the maximum signal propagation delay to a receiver on Earth is 87 ms [80]. During this time, a receiver's movement does not have a significant effect on the propagation delay. However, the much faster satellite movement has. Therefore, we compute the propagation delay at the transmit time t_i of a signal even though the receiver may still travel for an additional 87 ms.

The code phase $\phi_i(h)$ relates to the transmit time $t_i(h)$ of the received signal as follows:

$$\phi_i = t_i(h) \bmod 1 \text{ ms}$$

The transmit time $t_i(h)$ of the received signal at time h^t are related by the propagation delay $d_i(h)$ between the hypothetical position h^p and the satellite position.

$$t_i(h) = h^t - d_i(h) \tag{3.3}$$

The propagation delay can be found by dividing the spatial distance between the hypothetical position h^p and the satellite position p_i by the speed of light C:

$$d_i(h) = \frac{||h^p - p_i(t_i(h))||}{C} \tag{3.4}$$

The propagation delay $d_i(h)$ depends on the distance between the satellite position p_i at the time of transmission $t_i(h)$ and the hypothetical position h^p. The satellite position $p_i(t_i(h))$ at a given time can be computed from the ephemeris.

So the propagation delay $d_i(h)$ can be found knowing the transmit time $t_i(h)$ which itself can be found knowing the position of the satellite $p_i(t_i(h))$ which can only be found knowing the transmission time $t_i(h)$ for which the propagation delay $d_i(h)$ needs to be known. This circular dependency can be resolved by a short fixed point iteration which exploits the difference between the speed of light and the speed that the satellites travel with.

Namely, the signal propagation times from a satellite to a receiver on Earth range between 67 and 86 ms [80]. If we compute the signal transmit time using Equation 3.3 and this crude estimate we get $t_i \approx h^t - (67 + 86)/2$ ms $\approx h^t - 76.5$ ms The estimation error in the transmit time $t_i(h)$ is at most 9.5 ms. The maximum satellite speed relative to a receiver on Earth is 929 m/s [80]. This means that our estimate for $t_i(h)$ of 9.5 ms leads to a worst case satellite position estimation error of 9.5 ms $\cdot$ 929 m/s $= 8.83$ m. Using this new satellite position error, the second iteration starts with a new estimate of the transmit time $t_i(h)$, based on a satellite position error which is at most 8.83 m. Hence, the propagation delay estimation error is at most 8.83 m$/C = 19.4$ ns. The satellite position estimate that can be achieved using this propagation delay estimate already has a negligible error of 19.4 ns $\cdot$ 929 m/s $= 18$ µm.

3.3.3 Search Region

To guarantee the uniqueness of the solution, we limit the search region in which the set F of feasible hypotheses is contained. As GPS signals travel at the speed of light C, the C/A code phase of a satellite are the same for two hypotheses if their distances to the satellite differ by $k \cdot C \cdot 1\text{ms} \approx 300\text{km}$ for integer values for k. To avoid this affecting our results, we bound the search region in which the set of feasible hypotheses F is contained to a diameter of 300 km. Most likely the correct solution can still be found in larger areas, especially when more than four satellites are visible. The size of the search space follows a similar limitation. Note that the correspondence between time error and range error is given by the maximum relative satellite speed against a receiver, which is less than $1\frac{\text{km}}{\text{s}}$ on the Earth surface [80]. Hence, the search space in both space and time has to be small enough to make sure that the solution remains unique. For instance, a position range of 100 km and a time range of 50 km / $1\frac{\text{km}}{\text{s}} = 50$ s are guaranteed to deliver a unique solution.

For bounding the solution domain, one can use the antenna position of a cellular network as a reference. When the signal of the satellites is strong enough, we can also find the approximate receiver location with an idea presented by Liu et al. [49]. The authors show how the measured Doppler shift of a signal limits the receiver position to a cone. The position can be found at the intersection of the cones from each satellite. If we do not compute an initial fix, we can use the last computed position as an approximation for the new position.

3.3.4 Visible Satellites

The set V contains the indices of all potentially visible satellites. It is assumed to be the same for all hypotheses $h \in F$ and is the set of all the satellites with an elevation above the horizon larger than five degrees, as seen from the center of the search region. In theory, V is a function of a hypothesis h and the ephemerides, from which the elevation angles can be computed. However, it is safe to assume V is fixed with respect to all hypotheses since the elevation angles barely change within the search regions we consider. Also the Earth's rotation during the signal transmission can be neglected when computing the elevation angle of a satellite.

3.3.5 Space Discretization

The computation of the correlation values given in Equation 3.1 shifts the locally generated C/A code by its expected phase. In our case, the expected phase is rounded such that we shift by an integer value corresponding to one sampling interval T_s of the receiver. This helps to simplify the computation of the likelihood function as no signal interpolation is required. Due to the rounding, the likelihood of two hypotheses that are close may lead to the exact same set of C/A code phases ϕ_i for all visible satellites.

Ideally, we spread hypotheses in the search range such that no two hypotheses correspond to the same set of C/A codes to conserve computation resources. Also, we would like to have one hypothesis for every set of C/A code ranges which can be achieved within the search region.

Depending on the sampling interval T_s, we can compute the range difference that is required to change the value of the rounded C/A code phase ϕ_i. Namely, the corresponding "length" of a sample is $\lambda_s = T_s \cdot C$, where C is the speed of light ($\lambda_s \approx 37$ m for $T_s = \frac{1}{8\text{MHz}}$). Thus, for each satellite, the solution space is sliced into *spherical shells* with a slice width of λ_s. Each hypothesis in such a slice produces the same rounded expected C/A code phase ϕ_i.

With multiple satellites, the space is sliced in several directions as shown in Figure 3.2. This divides the solution space into volumes inside which all the hypotheses correspond to the same rounded C/A code phase and therefore equal likelihoods. Figure 3.2 shows a two-dimensional example.

Since we do not know the exact shape of the division of the search space in the volumes of equal observations, we sample the space with a regular grid. Ideally, this grid would be dense enough to "capture" all these volumes. However, some of these volumes can be infinitely small and thus, with any fixed grid density, we might not sample some volumes and therefore not find the most likely hypothesis. Luckily we can make sure that we do not miss

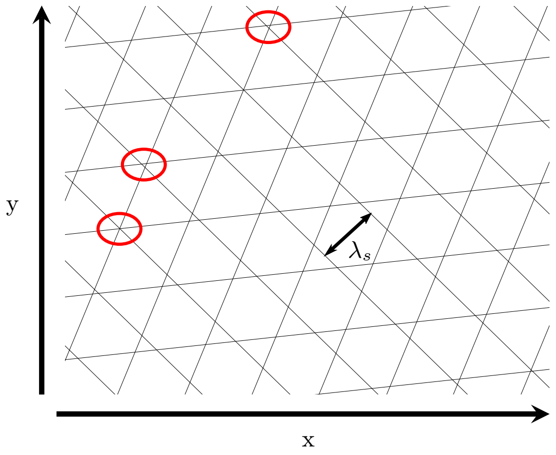
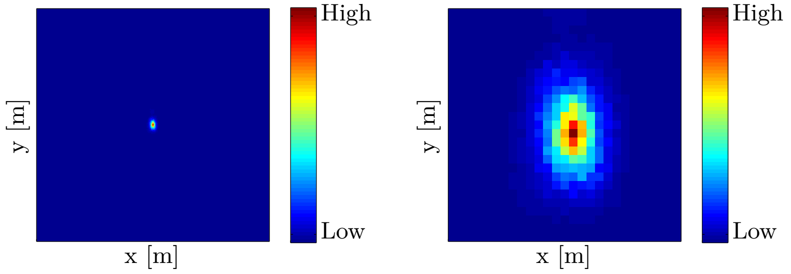


Figure 3.2: Two dimensional example of search space discretization. In this example, three satellites are visible which cause three groups of lines slicing the search space. When crossing a line, the expected C/A code phase ϕ_i for the corresponding satellite is rounded to the previous or next sampling period T_s. All positions inside a bounded area lead to the same likelihood. Very small regions may exist (indicated by the red circles).

the correct solution completely because no hypothesis is close enough by selecting the grid such that neighboring points are λ_s apart.

In this case, each hypothesis represents a cube of side length lambda. Such a cube has a diameter of $\sqrt{3} \cdot \lambda_s \approx 1.7 \cdot \lambda_s < 2 \cdot \lambda_s$. Since the space is divided into those cubes, an uncovered area can at most be half a diameter apart from the nearest hypothesis, that is the distance to the nearest hypothesis is less than λ_s. Note that a distance smaller than λ_s can at most cross one slice boundary for each satellite. This means that for an uncovered area and its nearest hypothesis the expected code phases ϕ_i are at most one sampling interval T_s apart.

The key observation is that our (and also common) GPS receivers oversample the GPS signals. For the correlation, this means that the peaks are not confined to a single sample of length T_s, rather their neighboring values are quite high as well and form a triangle-like pattern. Without noise, the correlation values at a distance of k samples from the peak have a value of at least $(1 - k \cdot 2 \cdot f_{\text{PRN}}/f_s)$ times the value of the peak. f_{PRN} is the rate of the PRN sequences (1.023 MHz). f_{PRN}/f_s is the fraction of the locally generated PRN sequence which does not match the correct part of the PRN in the signal. For a sampling rate of 8 MHz ($f_s = \frac{1}{T}_s$) for instance, the



(a) $\mathcal{L}(h)$ on a grid of 10 km by 10 km for a fixed time

(b) $\mathcal{L}(h)$ on a grid of 1 km by 1 km for a fixed time

Figure 3.3: In situations with line of sight between receiver and satellites the likelihood function is smooth and unambiguous. The figures shown are a cut through the search space where the time and height of the receiver have been fixed at the values corresponding to the most likely hypothesis. The distance between two points in the grid is approximately 37 meters.

directly neighboring values of the peak are at least 74 % as high as the peak itself, for a sampling rate of 56 MHz at least 96 %. Assuming the used sampling rate is at least 8 MHz, the found correlation values may at most be 26 % smaller compared to the largest one. Alternatively, we could filter the correlation values such that the correlation value at an index contains the highest correlation values amongst its direct neighbors. In this case, we are guaranteed to find the highest achievable likelihood, but the likelihood function is less sharp. We will discuss this idea further in Section 3.3.8.

3.3.6 Time Discretization

The hypotheses also have to be spread in the time domain. As in the spatial discretization above, we have to make sure that we sample densely enough, such that we do not miss the most likely position. If the hypothetical time for the correct location h^p is off by as few as $10 \cdot T_s$, its likelihood will be completely random (assuming T_s = 8MHz). This follows from the same argument about the shape of the PRN autocorrelation function above. In order to allow for sampling in the time domain, we exploit the fact that the expected C/A code phase $\phi_i(h)$ is approximately constant when varying the hypothetical time h^t by less than one ms. Hence, we simplify the computation of the correlation values $c_i(h)$ for hypotheses that are identical up to a difference in time t_μ which is smaller than 1 ms.

$$c_i(h, t_\mu) = \sum_{\tau=0}^{1\text{ms}} |r(\tau) \cdot \text{ca}_i(\tau - \phi_i(h) - t_\mu)| \tag{3.5}$$

We can simplify the computation of $c_i(h, t_\mu)$ for all $t_\mu \in [0, T_s, 2 \cdot T_s, \ldots, 1\text{ms}]$ using the correlation function C_i:

$$C_i(t_\mu) = \sum_{\tau=0}^{1\text{ms}} |r(\tau) \cdot \text{ca}_i(\tau - t_\mu)| \tag{3.6}$$

Note that the correlation function $C_i(t_\mu)$ can be computed independently of the hypothesis. By shifting the correlation function $C_i(t_\mu)$ of the received signal with the C/A code according to the expected phase $\phi_i(h)$, we can simplify the computation of the likelihood as follows:

$$\mathcal{L}(h) = \max_{t_\mu} \sum_{i \in V} C_i(t_\mu + \phi_i(h)) \tag{3.7}$$

This allows us to choose the time domain to be sampled at up to 1 ms intervals without letting a good solution be undetected.

In the worst case, an inaccurate time hypothesis shifts the most likely position by the highest possible speed of the satellites relative to the earth's surface ($1\frac{\text{km}}{\text{s}}$). This means that the localization error is expected to increase less than 1 m if the hypothetical time is off by 1 ms. Hence, we can further increase the intervals at which the time domain is sampled. This does not negatively affect the observations about the spatial discretization. We are still guaranteed to observe hypotheses that are very close to the highest achievable likelihood.

3.3.7 Averaging Likely Hypotheses

So far, we only discussed choosing the hypothesis with the largest likelihood as the solution. As described in Section 3.3.5, hypotheses that are near the correct solution should get a higher likelihood because of the triangular shape of the PRN autocorrelation function. See Figure 3.3 for an example. To improve localization accuracy, we consider the set of hypotheses H with the highest likelihoods. The set of most likely hypotheses is combined using a weighted average.

$$\bar{h^p} = \sum_{h \in H} \mathcal{L}(h) \cdot h^p$$

In Section 3.4 we discuss the performance impact of the averaging as opposed to selecting the location of the most likely hypothesis.

3.3.8 Efficient Implementation with Branch-and-Bound

Algorithm 3.4 Finding the n most likely points given a search space defined by a hypothesis h.

```
procedure S = GetMostLikelyPoints($n$,h)
    $n$: the number of likely points contained in S.
    h: the initial hypothesis defining the search space.
    $\mathrm{h}_{\mathrm{lmax}}$ = maxLikelihood( h )
    queue.add(h)
    S = $\emptyset$
    while queue.hasElement() do
        h = queue.popMostLikely()
        if $\mathrm{h}_{\mathrm{lmax}} \leq \min_n(\mathrm{h}_{\mathrm{lmin}} \in S)$ then
            continue
        end if
        $\mathrm{h}_{\mathrm{lmin}}$ = likelihood( h )
        $\mathrm{h}_{\mathrm{lmax}}$ = maxLikelihood( h )
        S.add(h)
        h[1] ... h[16] = splitHypothesis( h )
        for h[i] = h[1] ... h[16] do
            $\mathrm{h[i]}_{\mathrm{lmax}} = \mathrm{h}_{\mathrm{lmax}}$
            queue.add(h[i])
        end for
    end while
end procedure
```

Figure 3.3 shows horizontal cuts of example distributions of our likelihood computed from a one millisecond window of samples in good signal conditions. Our branch and bound method exploits that, under good signal conditions, large volumes in the solution space yield low likelihoods. The search space as discussed in Sections 3.3.3 and 3.3.5 is large as the hypotheses are spread at a distance of 37 m from each other and the search space spans 200 km $\times$ 200 km $\times$ 30 km. In addition to this, the time domain is searched within 10 s at intervals of 40 ms. This means that there are roughly $2 \cdot 10^{12}$ hypotheses which need to be tested. To reduce the number of hypotheses for which we need to compute the likelihood, we employ a branch and bound method as described in Algorithm 3.4. To do so, we need a method to compute both an upper- and lower-bound on the achievable likelihood (indicated by $\mathrm{h}_{\mathrm{lmax}}$ and $\mathrm{h}_{\mathrm{lmin}}$) within an area defined by a hypothesis . Note that in the algorithm, a hypothesis h contains the center position in x,y,z, and t as well as the size of the search space around it in all dimensions (x,y,z,t). The initial hypothesis covers the entire search space i.e. it extends over 200 km $\times$ 200 km $\times$ 30 km $\times$ 10 s. We approximate the lower bound of achievable likelihoods within an area as the likelihood of the hypothesis (h^p,h^t) which is indicated in the Algorithm as likelihood(h). For the upper bound (maxLikelihood(h) in the Algorithm), we use the expected code phases ϕ_i along with the size of the area covered by the hypothesis. Note that the larger the area covered by a hypothesis, the larger the uncertainty about the possible code phases ϕ_i. The uncertainty is given by the diagonal of the area covered divided by the speed of light. For a hypothesis with a diagonal of 10 km, the uncertainty is roughly 33 microseconds which corresponds to roughly 270 sample intervals T_s at 8 MHz.

To compute the upper bound on the likelihood, this can efficiently be taken into account. Instead of utilizing the correlation function as described in Equation 3.6, we apply a max-filter first.

$$C_i'(t_\mu) = \max_{\tau \in R} C_i(t_\mu) \tag{3.8}$$

R is the set of possible shifts that can be expected within the region covered by a hypothesis. In the example above with 10 km diagonal, $R = [-16.5\mu\text{s}, 16.5\mu\text{s}]$. The likelihood computation stays the same as in Equation 3.7 but using $C_i'(t_\mu)$ instead of $C_i(t_\mu)$. This yields an upper bound on the achievable likelihood inside the area covered by h. To further speed up the computation, the max-filtered correlation functions can be pre-computed as it is the same for all hypotheses covering areas of the same size.

Hypotheses in the queue are processed according to their maximum achievable likelihood h_{lmax} (popMostLikely()). This is crucial as areas with great potential are explored first, improving the likelihood that bad areas do not have to be further split up and analyzed. Each processed hypothesis is split in two in all dimensions (x,y,z,t) which leads to 16 new hypotheses, each covering a smaller volume of the search space. As soon as a hypothesis cannot achieve a higher likelihood than the n best hypotheses already observed, it s not further split up and discarded. The method guarantees that the n most likely points are found as only hypotheses are discarded which cannot possibly achieve a high enough likelihood.

Clearly, the performance of the algorithm depends on the quality of the received signal as the bounds will be more accurate for a smooth likelihood function. We will analyze the performance degradation as the signal quality deteriorates in Section 3.4.

3.3.9 Local Oscillator Frequency Bias

In practice, one of the problems we have to deal with is the frequency error of the local oscillator (LO). The LO in the front-end is not only used for the generation of the reference frequency for the frequency down-conversion, but also for the sampling rate of the ADC. Hence the error of the LO influences two parameters. First, the observed frequencies of the signals change. Second, the time that passes per sample changes. Akos [1] states that the frequency for the locally generated C/A code should match the actual frequency with an error smaller than 250 Hz. Otherwise correlation peaks are hard to find even under good signal conditions.

During the acquisition phase in conventional receivers, the Doppler shift of each satellite is estimated by correlating the received signal with multiple frequency shifted versions of the C/A code. The frequency shifted C/A code which matches the received signal best is used to estimate the arrival time and also gives information about the sum of the LO offset and the Doppler shift. After the acquisition, the Doppler shift and hence the LO, is known only approximately to reduce the computational complexity during acquisition. This approach could be replicated in our solution to estimate the LO offset.

Similar to the search performed in classic receivers, we could track the LO offset by computing the C/A code correlation functions for different frequency offsets. Note that since we do compensate for the Doppler shifts using our prior knowledge, we only need to estimate the LO offset instead of the sum of the LO offset and the Doppler shifts of each satellite.

In our test setup described in Section 3.4, we measured the LO offset initially using the classical GPS approach. Over the course of 1.5 years, all experiments were performed with the same, constant LO offset (+1.9ppm). Careful calibration of the LO can reduce the impact of its errors to an extent that is acceptable. The performance details are discussed in Section 3.4.

3.4 Evaluation

For the evaluation of our method, we used an Ettus USRP B200 software radio with a standard GPS patch antenna from Trimble Navigation. Samples were recorded as 8 bit I/Q samples with 8 MHz sampling frequency. We made recordings of several minutes and cut out windows with one millisecond length every 0.999 seconds. We did not choose exactly one second, since bit flips in the navigation signal, which severely degrade the signal quality, can occur every 20 milliseconds. To prevent these to always have an influence on the same satellites' signal, we chose a slightly shorter interval.

We used navigation data originally broadcast from the satellites, which we downloaded from NASA's archive of space geodesy data[4] [62]. For the time synchronization, we determined the time of the first sample received from the RF front-end with the Network Time Protocol (NTP). The start time of subsequent one millisecond windows was estimated by counting the number of elapsed samples in the recorded data stream.

To evaluate the accuracy of our algorithm, we placed the receiver antenna on a survey point located on our university building. The location of this point is known accurately. We expect errors in its location to affect our results negatively giving us a lower bound of the performance.

Unless otherwise indicated, experiments were performed under good signal conditions (direct line of sight to most satellites above the horizon) and the search space size was 200 km $\times$ 200 km $\times$ 30 km $\times$ 10 s. The reason for the size of the search in the time dimension, is that with a good oscillator with maximum drift of 0.5 ppm and an initial time error of 50 ms (easily achievable with NTP), a range of ± 5 seconds covers a duty-cycle interval of up to 114 days. So, 10 seconds are a very large bound on the time search especially since time inaccuracies can be compensated when a fix is computed.[5]

For each processed one millisecond window of signal, we varied the grid of hypotheses uniformly at random in each dimension, up to half the distance between two points. This eliminates possible bias from a specific positioning of the grid. For instance, if one hypothesis always matched the correct receiver position and time exactly, the results might look much better than if the correct position and time lie in the center between the closest hypotheses in each dimension.

3.4.1 Averaging Over Likely Hypotheses

First, we evaluate how the accuracy depends on the number of most likely points used to compute the weighted average as described in Section 3.3.7. Figure 3.5 shows the cumulative distribution functions of 501 fixes covering approximately 500 seconds with the duty cycle of 0.999 s. The shown numbers of points are $\{1, \ldots, 7\}$ to the power of four, since we expect the points around the correct position to have the highest likelihoods. So, the idea is that the curves show the results when averaging over the hypercubes in four

[4]We used the "Daily GPS Broadcast Ephemeris Files" data set that can be found at `http://cddis.nasa.gov/Data_and_Derived_Products/GNSS/broadcast_ephemeris_data.html`

[5]We think a more reasonable upper bound on the duty-cycle interval would be one day, which means that with such a large time search, we could tolerate many failed localizations between two successful ones, for instance when the receiver is indoor for a long time period.

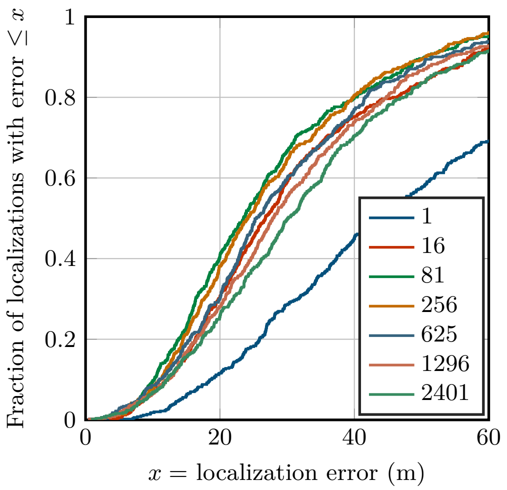


Figure 3.5: Accuracy of our method with different numbers of most likely points used for the weighted averaging. Cumulative distribution functions of positioning error (distance to ground truth).

dimensions with side lengths of one to seven hypotheses around the correct position.

The best accuracies are achieved with 81 or 256 points. Since lower number of points correspond to a higher likelihood threshold to eliminate regions of hypotheses with low maximum likelihood (see Section 3.3.8), we use 81 points in the following, as this will save more computation time.

Note that existing CD methods search for the best point only, which is clearly suboptimal. The median position error with 81 points is 23.5 m, which is almost twice as good as the solution with the best point only, which has a median error of 44.3 m. The standard deviation is 17.3 m with 81 points and 27.6 m with the best point only. This shows that our weighted averaging is a substantial improvement over standard CD, substantially improving accuracy.

3.4.2 Position Averaging over Time

To understand the trade-off between accuracy and the amount of data used, we tested the influence of averaging multiple positions computed from different one millisecond long windows (sliding window average). The results – obtained again from 501 windows – are shown in Figure 3.6. With just a

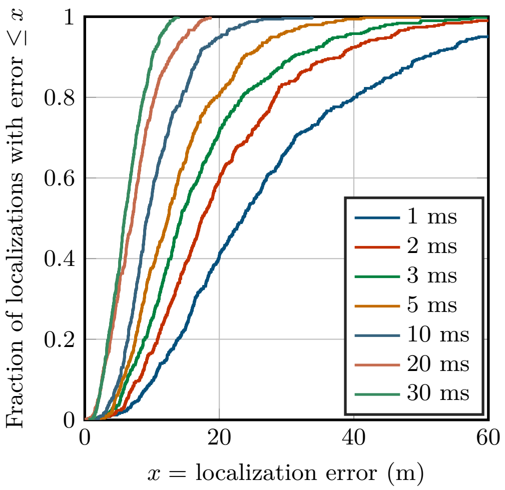


Figure 3.6: Comparison of positioning accuracy when averaging over different numbers of consecutive fixes.

few more milliseconds, we can gain significant accuracy. For instance, with two milliseconds of data, the median positioning error drops from 23.5 m to 17.4 m. With 10 ms, it even drops to 9.2 m. And with 30 ms, *all* positions are within 13.9 meters.

3.4.3 Horizontal Positioning

To evaluate the accuracy when searching in space only horizontally, we fixed the altitude for the search to that of the ground truth. This emulates scenarios where the receiver is 1) on the Earth surface, so the height can be determined using an Earth elevation model (for example the United States Geological Survey (USGS) elevation model[6]) or 2) the receiver has a barometer, whose measurements can be used together with meteorological data to determine the altitude. The benefit of such an approach is not only better accuracy, as can be seen in Figure 3.7, but the search space is reduced by one dimension, resulting in less hypotheses to test, which translates to faster and less energy consuming processing. For the positioning with fixed height, we also first determined the best number of points for weighted averaging

[6]More information about the USGS elevation model can be found at the "The National Map" website: `http://nationalmap.gov/elevation.html`

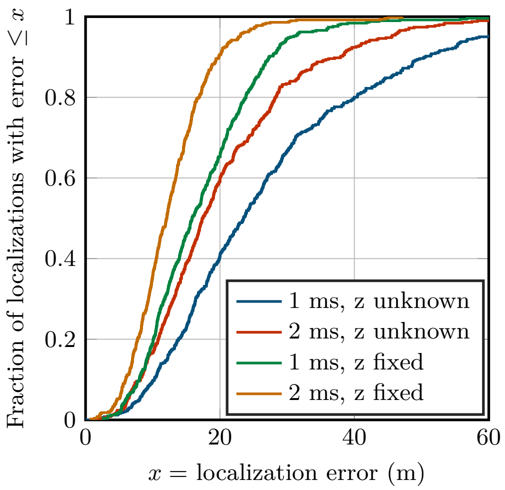


Figure 3.7: Positioning accuracy with and without fixed height.

with the same procedure as explained in Section 3.4.1, although with numbers to the power of three, because the search space is three dimensional. The best number of points turned out to be 64. Also for this experiment, the number of one milliseconds windows processed was 501.

The idea of using an Earth elevation model to restrict the possible solutions has also been used by Liu et al. [49]. Because we do not have an implementation of CTN available, we cannot directly compare our results to theirs. However, the box plots in their paper show a median error of approximately 40 m with 2 ms of data used. Our median error when using 2 ms of signal and fixing the height of the solution is 12.1 m. This suggests that our approach is competitive.

3.4.4 Computation Time

To show how the performance of our method using branch-and-bound depends on the signal conditions, we conducted two experiments capturing both very good signal conditions (rooftop) as well as very bad signal conditions (inside a multistory university building). We reduce the search space to 10 km × 10 km × 1 km × 4 s for this experiment, to also be able to test the brute force implementation which tests every single hypothesis.

Figure 3.8 shows the cumulative distribution functions in both indoor

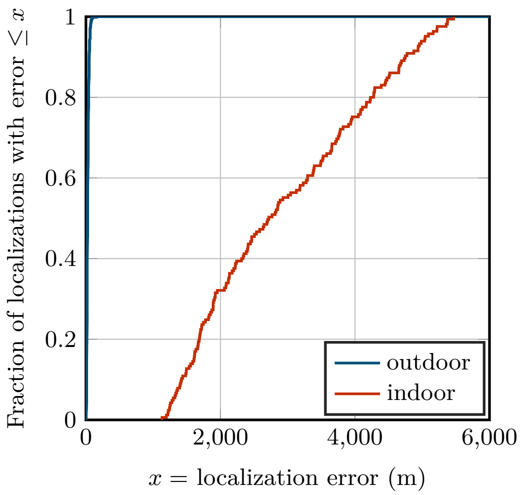


Figure 3.8: Positioning accuracy with different signal qualities. Outdoors the solution is much more accurate than indoors.

and outdoor scenarios as described in the last paragraph. It clearly shows that the indoor scenario did not allow for a meaningful localization.

For the indoor test, the computation time was 240s whereas in very good conditions, the time is only 18.6 s. Note that the indoor test presents a worst case scenario in both computation time and localization accuracy. The brute force implementation takes more than two hours to complete in any scenario.

This means that even in situations that make it difficult to find a fix, we find the most likely location in reasonable time compared to a brute force implementation. For the previous experiments with the larger grid in good signal conditions, our method takes 31s of computation time.

The performance corresponds to the execution on a current Intel i7 mobile processor with a single thread. The runtimes are not indicative of an optimized implementation of our method, since it could easily be parallelized because the computation of the likelihood is independent for each hypothesis. In all the above experiments about computation time, roughly $1.5 \cdot 10^3$ hypotheses are evaluated each second. A working CUDA implementation of the brute force method revealed that on a Nvidia GTX 1080, roughly $3 \cdot 10^6$ hypotheses can be evaluated each second which indicates that the search can be sped up 200 times.

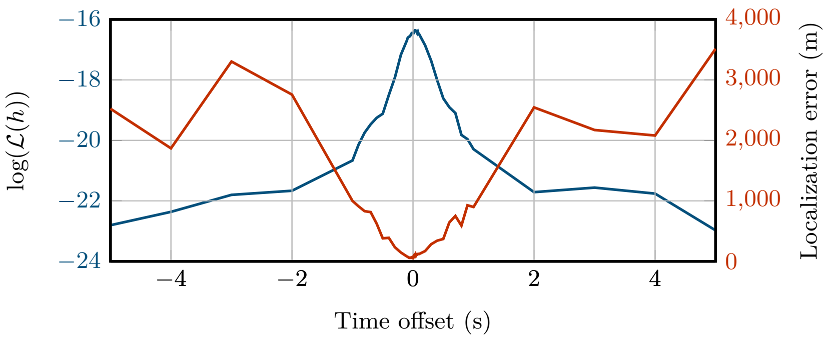


Figure 3.9: Shape of $\mathcal{L}(h)$ for large time errors.

3.4.5 Time Dependence of the Likelihood Function

To test the influence of the time parameter in our likelihood function, we picked a random one millisecond long window of the sampled signal and searched the position which maximizes the likelihood given different receiver times. The results are shown in Figures 3.9 and 3.10, other ms windows exhibit the same properties as described below. The plot on top shows that, at least for signals with good quality, our likelihood function (in blue) is roughly convex in the time dimension. However, we cannot reconstruct the correct time precisely because the probability of the best point does not change significantly when the time is within a second of the correct time (blue curve in the right hand side plot). However, the positioning quality does vary significantly inside this time range (red curve in the right hand side plot). This is due to the fact that within the search space, there are points which still match the received signal well. Judging from the localization error, the most likely position passes the correct position in a linear fashion as the time error is varied from negative to positive. This suggests that the likelihood function is quite flat in the time domain which is one of the reasons why the averaging over the most likely hypotheses helps to increase the accuracy of our method.

3.5 Conclusion

We have shown how Collective Detection can be optimized to compute a location fast, even from coarse initial guesses about both position and time. Our branch and bound method scales well in both good as well as bad signal conditions. The localization performance is superior to similar approaches

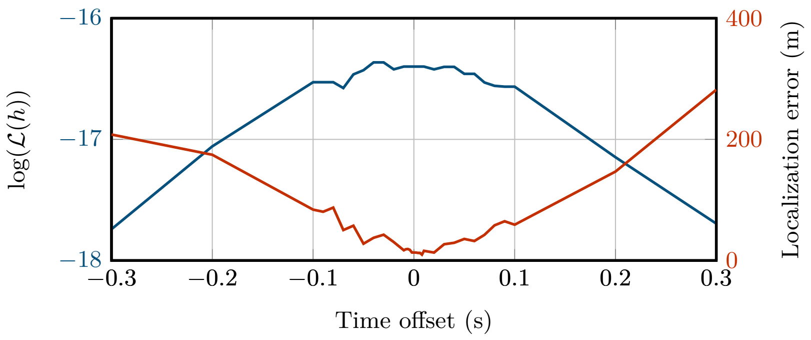


Figure 3.10: Shape of $\mathcal{L}(h)$ for small time errors.

due to the averaging which greatly reduces the effect of the flatness of the likelihood function in the time domain. When utilizing more than one millisecond of signal, the performance is very competitive even with classical GPS receivers consuming much more energy.

A work in progress hardware prototype indicates that a tracking device running off of a coin-cell battery could record one one ms signal sample every hour for as long as two years. Since our receiver does not require accurate time synchronization and the footprint of the PCB can be reduced to match the coin-cell which is a great advantage over the recent work of Liu et al. [49] which used a large and heavy DCF-77 receiver for time synchronization. For connected devices such as smartphones, all the computation can be performed in the cloud which frees the mobile device from the computational burden completely.

4 Touch in Motion

We have seen how mobile devices can observe their environment to deduce their location. Motion sensor helped us to learn about environmental properties such as WiFi signal strength distributions. The utility of smartphones with accurate location and environmental sensors means that people carry such devices in most day to day situations. This opens opportunities for violations of privacy.

These sensors, on a first glance, do not reveal sensitive information. Therefore, in past and current versions of Android and iOS, they can be accessed by any application installed on the device, even by applications running in the background. For security reasons, the same does not apply for touch screen input. Only foreground Apps are granted access to touch input data since it reveals, among other things, characters being typed on the on-screen keyboard which may include passwords or private information in text messages. In most situations, when touching the screen of a phone, the phone will move. Depending on the usage scenario this motion might be very small and hard to track, for example when the phone is lying on a table. However, usually interaction with hand held mobile devices causes the device to move so much that the built in accelerometer and gyroscope sensors are able to track this motion. The relation between touch and motion raises the question: how well one can infer touch from motion input? Being able to do so presents a security threat for most of today's smartphones.

Mobile devices are by design used in a large variety of environments that directly influence the motion sensor readings. This drastically complicates the task of inferring touch from motion and therefore has to be taken into account when evaluating any inference mechanism. To build a data set that reflects variable real world environments, we collect data through an Android game. Players are not instructed to hold or interact with the device in a specific way. Therefore, we do not know or take into account if a player is sitting in a train or walking while playing.

The player's task is to memorize and imitate patterns that mimic the lock-screen patterns found on most current mobile operating systems.

We describe how an attacker could collect both touch and motion data to train a classifier, that can be used to derive touch input in any application being used on the device under attack. We use the Dynamic Time Warping (DTW) algorithm to compare and classify gestures and evaluate classification accuracy for touch inputs. By performing this side channel attack, an attacker can steal passwords or at least reduce the number of guesses required to do so. Our results are based on a large scale user study that covers arbitrary and unknown usage scenarios, 1'493 users and 615 of different Android device models. We show that varying environmental influences impact performance heavily.

4.1 Related Work

Motion sensors have previously been used for side channel attacks. In most cases data was recorded in controlled environments, thereby reducing the impact of real world effects such as varying user activities. An attacker cannot be sure that motion data was recorded when the user was sitting still, instead the user might be walking or riding a train.

Examples of such results include the paper by Cai and Chen [23], who showed that side channel attacks on touch input using motion sensors are feasible in a lab environment. Aviv et. al. [5] collected data from 26 participants while sitting or walking. Although this study helps understanding the effects of added disturbances, the environment is still controlled and known. The study with the largest data set has been performed by Cai et. al. [24]. They collected 47'814 keystrokes from 21 test persons, with 4 devices in a lab setting. Miluzzo et. al. [57] performed a study comparing multiple classification algorithms with data from 10 test persons while sitting or standing. Although their results identify how to best infer touch from motion input, their data set also limits the applicability of their results in the real world. The work of Xu et. al. [87] focuses on tap gestures and describes a game which collects training gestures when running in the foreground and test gestures when running in the background. However, their data set spans only three users and was recorded in a controlled environment. Owusu et. al. [64] focus on random forests to detect passwords on smartphones by analyzing the accelerometer readings of 4 test persons. Other studies [45, 88] show that smartphone users can be distinguished based on their touch and motion input behavior. Hinckley et al. [39] show that the combination of touch and motion data can be used to create novel ways to interact with our mobile devices.

In contrast to the papers described above, we focus on collecting data in uncontrolled environments. Namely, our data set originates from users that were primarily concerned with playing a game on their smartphone and were not instructed about how and where to do so. The environments may range from office spaces to airplanes or trains. Since our data acquisition process can be replicated by an attacker, our data set allows us to assess how realistic a side channel attack on touch input really is. We collected data from 1'493 test users, which generated more than a million gestures on 615 distinct device models. This is roughly 50 to 200 times more participants and up to 20 times more gestures than in previous studies. To our knowledge, there is no similar work with the same magnitude of collected data in similarly unconstrained environments.

Touch input was not the only target of side channel attacks using motion sensor data. Liu et. al. [48] tried to infer three-dimensional, free-hand movements using accelerometer readings. They collected 4'480 gestures from 8 test persons over several weeks. With (sp)iPhone, Marquardt et. al. [53] developed a mechanism to infer input on a physical keyboard by analyzing the motion sensor readings of a phone laying next to the keyboard. The proof of concept Android application Gyrophone [56] demonstrates that it is possible to recognize speech using a gyroscope sensor. Niu et. al. [61] used Dynamic Time Warping to measure the similarity of gestures to authenticate users. Recognizing ten distinct gestures, Chong et. al. [28] used the motion sensors to unlock the phone by performing a user defined series of gestures. Since the gesture detection is performed independently of the user and the raw information is not used in the authentication process, this is very similar to a password composed of ten letters that are entered through performing gestures instead of pressing keys.

4.2 Method

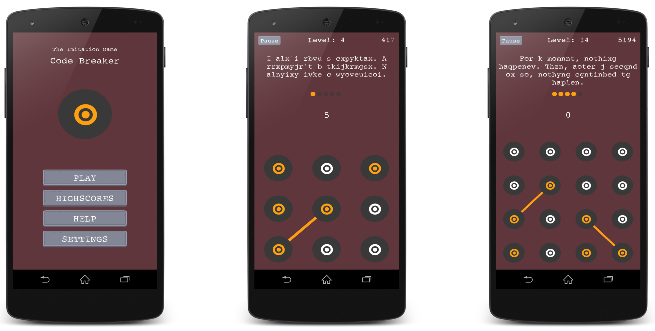
To simulate a realistic attack scenario, we decided not to invite test persons into a test laboratory with a predefined and controlled environment. Instead, we developed an Android game, which we distributed on the Google

Play store[1] to collect data. The same method can be used by an attacker and might already be exploited. In the game, the player's task is to memorize and reproduce patterns on the screen as shown in Figure 4.1. The patterns are displayed in the bottom half of the screen, where one usually finds the keyboard or pin input field. As the levels get harder, the grid resolution is increased from 3×3 to 4×4 touch elements, which we call cells. The game not only asks the user to press specific cells in this area, which we call tap gesture, but also to connect cells using swipe gestures. In contrast to an attacker, we informed users upon installation and first launch of the game that motion data is collected for research purposes and only when the game is running in the foreground. Touch input is measured in terms of x,y coordinates or a series of them for swipe gestures. In addition to touch input, the game also records the x,y,z coordinates of both accelerometer and gyroscope sensors built into the device. All recorded data is linked to a randomly generated unique id that is generated when the game is first installed. Since different users might play the game on the device, this unique id is device (and not user) specific. In addition to this, we collect basic device information such as the device manufacturer and type. When connected to a WLAN, the compressed log files are sent to a central database. When analyzing users with bad classification performance, we observed that their motion sensor measurements contain random readings close to zero. This indicates, that the device might not be moving enough, possibly being placed on a table or otherwise fixed. We excluded such users from our experiments. The number of users evaluated for each experiment are mentioned in the respective sections.

4.2.1 Preprocessing

The motion data is first segmented using the touch input as ground truth. To account for device motion that occurs before and after the screen is touched, we leave a variable amount of sensor readings before the touch gesture starts and after it ends. The time window is at most 100 ms. This segmentation can also be performed in an attack scenario, at least for the training data collection phase, using the same technique. For the classification task, we can use device events to segment motion data. For example, power on and unlock events provided by the operating system can be used to segment pattern or pin inputs performed to unlock the device. To remove the effects of gravity and gyroscope drift, we remove the mean values for each individual sensor axis. We thereby implicitly assume that the device orientation and velocity is the same at the beginning and at the end of each gesture.

[1]Game on Google Play. `https://play.google.com/store/apps/details?id=ch.ethz.pajonas.ba.imitationgame.android` (2015-03-13)



I alx'i rbvu s cxpyktax. A rrxpmyjr't b tkijkrmgsx. N alnyixy ivke c wyoveuicoi.

For k momnnt, nothixg haqpenev. Thzn, aoter j secqnd ox so, nothyng cgntinbed tg haplen.

Figure 4.1: Screens of the game showing the main menu, and both the 3×3 as well as the 4×4 grid the player interacts with during the game.

4.2.2 Classification

We do not attempt to guess the exact pixel the user touched, but rather predict the cell, which was pressed or swiped by the user. We build a classifier for each separate unique id and assume an id to relate to one single player although multiple users might play the game on the same device. To quantify the similarity of two gestures, we use the *DTW* algorithm [59] with varying cost functions to compare individual time samples as described in Section 4.2.3. The user model M consists of 10 motion sensor recordings for each cell i on the grid M_i. In order to classify a test gesture t, our algorithm computes the *DTW*distance to all samples in M and selects the cell with the minimal *DTW* distance (see Equation 4.1). We also evaluated different metrics, such as the average and median *DTW* distance (Equation 4.2).

$$\text{class}(t) = \arg\min_{x} \min_{i \in M_x} \text{dtw}(t, i) \tag{4.1}$$

$$\text{class}(t) = \arg\min_{x} \left(\frac{1}{|M_x|} \sum_{i \in M_x} \text{dtw}(t, i) \right) \tag{4.2}$$

However, the average *DTW* distance is not robust against outliers in the training set, and therefore was expected to produce worse results than the min and median distances. Alternative classification techniques used in similar work are Hidden Markov Models [5] or feature based approaches, Support Vector Machines [84].

4.2.3 Dynamic Time Warping Cost Functions

To perform the time series analysis using the DTW algorithm, we need to select features, which we can use to describe the distance or matching cost between two sensor events from two different gestures. Each motion sensor event consists of 3 accelerometer and 3 gyroscope measurements, one for each spatial dimension. These 6 coordinates form a feature vector $A = \{\text{acc}_x, \text{acc}_y, \text{acc}_z, \text{gyro}_x, \text{gyro}_y, \text{gyro}_z\}$, representing the values of the x, y, and z accelerometer and gyroscope coordinates of a single sensor event in a touch or swipe gesture. One possible metric to calculate the distance between two sensor events is the Euclidean distance, as Niu et al. suggest in their work [61]. Cai et al. [24] propose a different feature, calculating the two-argument arctangent (atan2) using the x and y axis of the accelerometer readings, arguing that motion data on the z axis is not a good feature to infer keystrokes (see Equation 4.3). We designed a new metric which pairwise calculates atan2 for all accelerometer and gyroscope axes combinations and then sums up their absolute differences (Equation 4.4). The equations below show how those three metrics are used to compare two sensor events i, and j in two different gestures A and B. Variations of the metrics above and others like the Manhattan distance or the L_∞-norm are also included in our experiments. The Manhattan distance is the sum of the accelerometer and gyroscope differences between two measurements.

$$c_{acc}^{xy}(A_i, B_j) = \left|\text{atan2}(A_{acc}^{i,x}, A_{acc}^{i,y}) - \text{atan2}(B_{acc}^{j,x}, B_{acc}^{j,y})\right| \tag{4.3}$$

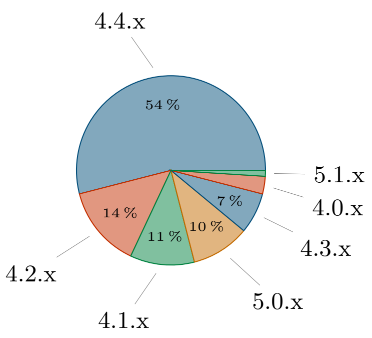
$$c_{acc}(A_i, B_j) = c_{acc}^{xy}(A_i, B) + c_{acc}^{xz}(A_i, B_j) + c_{acc}^{yz}(A_i, B_j)$$

$$c_{sum}(A_i, B_j) = c_{acc}(A_i, B_j) + c_{gyro}(A_i, B_j) \tag{4.4}$$

4.2.4 Dynamic Time Warping Penalty

In order to penalize sequences whose time axis needs to be stretched a lot, we employ different penalization factors p. See Equation 4.5 for the recursive definition of an entry in the DTW matrix $d(i, j)$. In the penalization experiment, we use the c_{sum} distance metric to compare two measurements A_i and B_j at times i and j respectively. By increasing p, the cost of advancing time i and j unevenly is penalized. This leads to a higher DTW matching cost for sequences of uneven length or speed.

$$d(i,j) = \min\{ \begin{array}{ll} d(i-1, j-1) & + c_{sum}(A_i, B_j) \\ d(i, j-1) & + c_{sum}(A_i, B_j) \cdot p \\ d(i-1, j) & + c_{sum}(A_i, B_j) \cdot p \ \} \end{array} \tag{4.5}$$



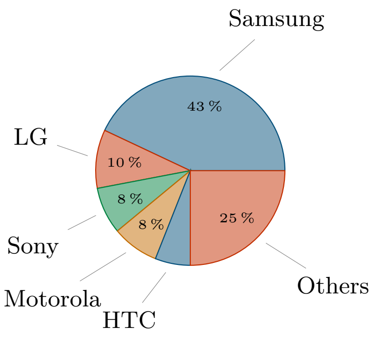


Figure 4.2: Variety of the devices and Android versions on which we collected our data set. Data has been collected from 615 distinct device models from 25 manufacturers with over 38 different screen sizes.

4.3 Data Set

At the end of the 4 month data collecting phase, the game reached 2’049 installations. Most of the users are from India, USA, Italy, or Switzerland. The Android application received 70 ratings in the Google Play Store, with an average rating of 4.01 stars out of 5. The most used device to play the game is the Google Nexus 5. Figure 4.2 shows that most players’ phones are produced by Samsung, and the most popular Android version is 4.4. Data has been collected from 615 device models with over 38 different screen sizes. The server application received data from 1’493 users, who played a total of 87’962 levels. With an average of 15 gestures per level, this corresponds to more than a million collected gestures. Since the data has not been collected in a laboratory, it contains unknown external influences on the motion data, devices with malfunctioning motion sensors, and users with very few gestures.

4.4 Evaluation

A training gesture is a gesture for which we know the motion sensor readings as well as the ground truth touch data, which we collected with our Android application. A test gesture on the other hand, is a gesture for which the touch input is ignored and the task is to infer the correct touch gesture using only the motion sensor readings. Only users with working accelerometer and gyroscope sensors are included in the experiments. Not all players generated enough data to generate a model for each cell. In these cases, we only classify

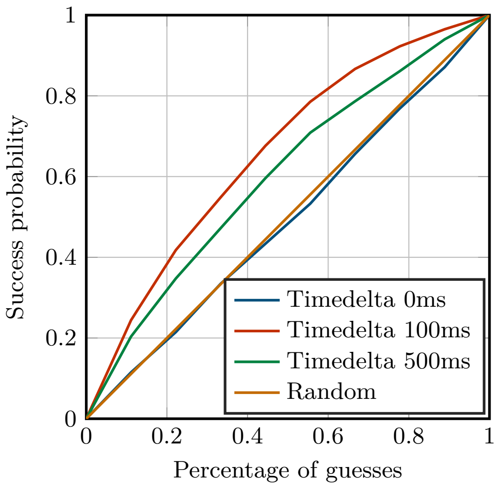


Figure 4.3: Comparison of different segmentation time windows. A larger window means that there are more pre- and post- measurements. In this plot, tap gestures on the 3×3 board from 14 users have been evaluated

cells with at least 10 recorded training gestures. Hence, a user might have fewer than 9 or 16 cells with enough training data. The random guessing probability is adapted to compensate for the reduced solution space of the classification problem. To generate enough training data for every single cell, a user is required to interact with the Android application for about 30 minutes. Depending on the experiment, between 10 and 500 players were used to evaluate performance changes.

4.4.1 Segmentation Time Window

The segmentation time window controls the number of pre and post measurements of each gesture. This window can be varied to optimize the classification performance since the device moves already before touch input is registered. For this experiment, we use our sum of atan2 cost function as described in Equation 4.4. The DTW penalty is set to 240% and we use the min classification method as described in Equation 4.1. Figure 4.3 shows different time windows of the length 0 ms, 100 ms, and 500 ms. Including no measurements that are recorded before the touch gesture starts produced bad results. This leads to the conclusion that the motion of the device before the touch event actually starts is the most discriminant. The time window

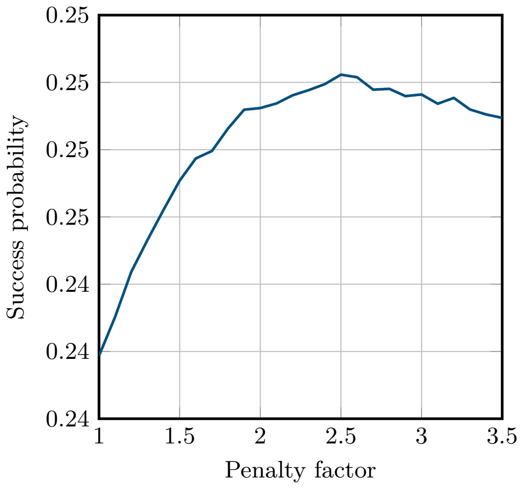


Figure 4.4: Influence of different penalties on the classification performance. In this experiment, all users have been evaluated for tap gestures on the 3×3 board. We have evaluated the probability of correctly guessing the label with the first attempt.

of 500ms also performs significantly worse than 100ms, which is why we chose 100ms for all the other experiments. In our work, we do not focus on the segmentation of the test gestures. To detect the unlock pattern, it seems to be sufficient to trigger the measurements using the unlock events of the smartphone. To infer PIN entries or multiple gestures, one would need to segment the test gestures. According to previous results [24, 87], there exist promising segmentation methods to achieve this.

4.4.2 *DTW* Penalty

To evaluate the effect of varying *DTW* time penalties, we analyzed the classification results from all players on the 3×3 grid utilizing the sum of atan2 metric as described in Equation 4.4 and the min classification method as described in Equation 4.1. We evaluate the penalty in the range of $p = 100\%$ to $p = 350\%$ in 10% increments. Figure 4.4 shows the effect of a varying *DTW* penalty. As expected, penalties that are very small or large result in worse performance. For very small penalties, sequences can be stretched beyond what is to expect due to the natural variance each user causes. In case of very high penalties, the *DTW* algorithm mostly

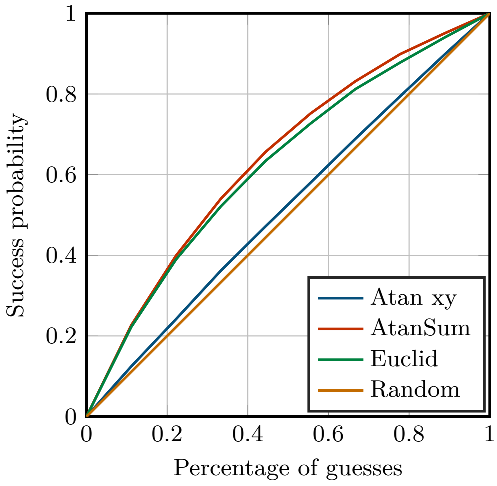


Figure 4.5: Comparison of the distance metrics used in the *DTW* algorithm (described in Section 4.2.3). The sum over all atan2 features performs best and is used as the distance metric for the other experiments. In this plot, all users have been evaluated for tap gestures on the 3×3 board.

matches sequences without stretching either time axis, resulting in a score that is very close to the sum of all corresponding sample costs. The best classification results can be achieved with p at 240% but as one can see, the performance differences are marginal for non extreme choices of p.

4.4.3 *DTW* Cost Functions

Figure 4.5 shows the comparison of the four presented distance metrics in the *DTW* algorithm. For this experiment, all users were evaluated using a *DTW* penalty of 240%. Our proposed sum over all atan2 features metric performs slightly better than the atan2 function and is therefore chosen as the distance metric in all other experiments. The euclidean distance function performs worse than both of the other distance metrics, but still much better than other metrics we tested (Manhattan, L_∞) which we omit on this plot.

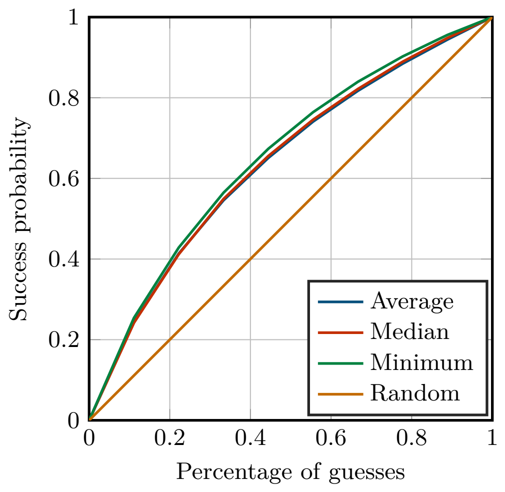


Figure 4.6: Comparison of the classification methods minimum, mean and median as described in Section 4.2.2. The minimum metric performs best and is used for the other experiments. In this plot, all users have been evaluated on the 3×3 board.

4.4.4 Classification Methods

The performance comparison for the presented classification distance metrics for all users on the 3×3 grid is shown in Figure 4.6. The DTW penalty was set to 240%. Interestingly, performance variations are insignificant for all three methods. Since our training sets may include outliers, we expected the average method to perform significantly worse than the min and median method, respectively. If the user e.g., bumped into someone or walked up steps while performing the tap gesture, then this outlier will skew the results and make the classification task more difficult. The minimum distance method performed best in this experiment and is therefore used in the other evaluation tasks.

4.4.5 External Influences

To evaluate the impact of changing environments, we compare the performance of heavy- and light-users while using a fixed training set of 10, a DTW penalty of 240%, the min classification method (Equation 4.1), and the sum of atan2 features. Heavy users played more levels and hence, produced more data. We expect their data set not only to be bigger, but also to

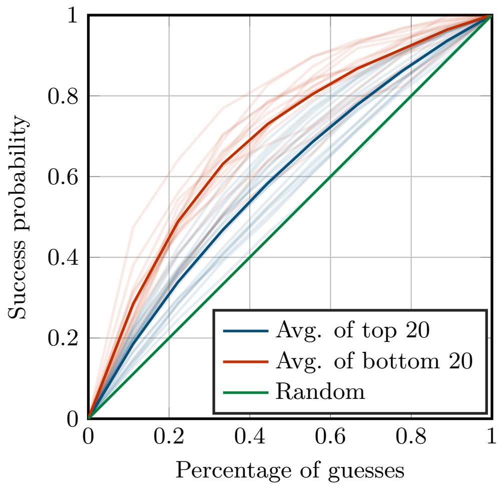


Figure 4.7: Performance comparison between heavy- and light users with fixed training set size. The environments in which heavy users played the game cannot be captured by a small training set size. Therefore, small training sets are sufficient, as long as the environmental effects are similar in both the training and testing phase.

contain more varied environments. Since we limit the training set size to 10 for both user groups, we expect the performance for heavy users to be bad since the small training set cannot capture all environments the game was played in. Figure 4.7 shows the 20 users with the most collected gestures (heavy) in blue and the 20 users with the least gestures in red (light) out of more than 200 users in total. Guessing the correct cell in the first try for the bottom 20 users is with 29% roughly 10% higher than for the top 20 users (19%).

4.4.6 Training Set Size

The results in the previous section beg to evaluate the same two user groups while using larger training sets for the heavy users. Performance for heavy users should increase as the training set captures more environments. To evaluate the effect of varying training set size, we lifted the restriction to only use 10 and instead trained our model with as all available samples for each user. This means that only the test gesture is excluded from the

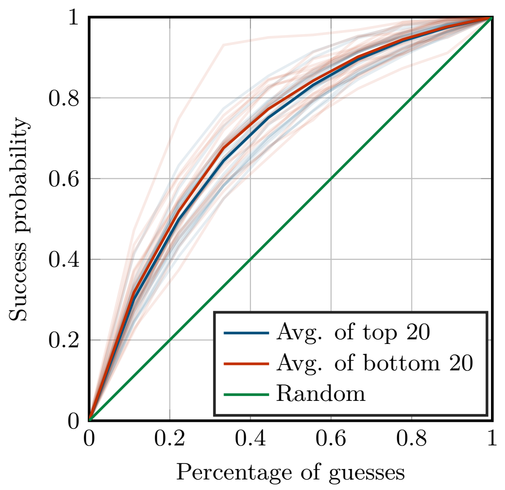


Figure 4.8: Impact of varying training set sizes on the classification performance. In this experiment, all training samples were used for each user on the 3×3 grid. Users with the most training data (in blue) show similar performance as compared to the users with the least training data (in red).

training set. Thereby we remove the advantage of light users being able to capture a larger fraction of the environmental effects in the training set when compared to the heavy users. In our data set this means that the training set can be up to two orders of magnitude larger. Classification was performed using the min metric (Equation 4.1). Figure 4.8 shows the 20 users with the most collected gestures in blue and the 20 users with the least gestures in red (out of more than 200 users in total). The classification rate of the top 20 users differs insignificantly (1.5% on the first guess) from the bottom 20.

Since both heavy- and light-users are tested using training sets that capture all environments the game was played in, performance is consistent for both user groups. For an attacker, this means that small training set sizes are sufficient, as long as the environment under which touch input inference should be performed is similar to the one predominant during training data set collection.

Both the experiment on the training set size, as well as the one on the environmental effects were performed using the exact same two user groups. As long as the training set captured the external influences affecting the

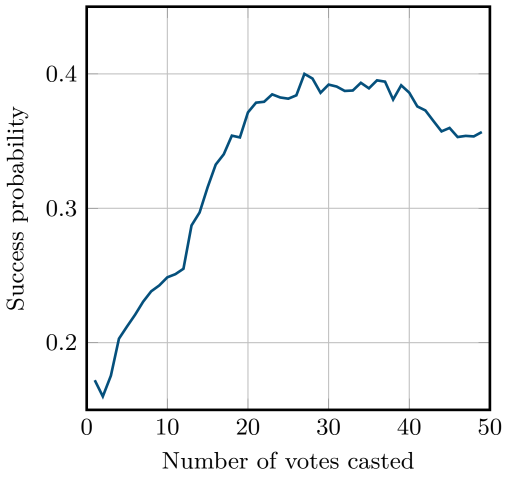


Figure 4.9: The probability of guessing the correct label for a given set of test gestures improves from less than 20% to more than 40% when increasing k from 1 to 27 test gestures

classification phase, performance is insensitive to the size of the training set. Collecting a large training set therefore helps capturing more environmental influences but does not allow the classification accuracy to improve significantly once an environment has been captured.

4.4.7 Repeated Attack

In this section we try to emulate an attack on the user's pin code to unlock the screen. Since users enter the same pin code over and over again, an attacker could use multiple motion measurements to improve the guessing accuracy. We know that after turning on the screen, the first number entered is always the first digit of the pin code. Thus, instead of using one test gesture, we use several of them. The task is now to classify these gestures, about which we know that they belong to the same label, but not to which one. The simplest approach is to cast a vote for each test gesture's most likely label according to the previously described method and then pick the label with the highest number of votes. Note that we limit our setting to k gestures in order to evaluate the influence of the number of votes on the classification accuracy we can achieve. If a user has more than k gestures, we limit it to k artificially and we keep the training set size fixed to 10.

As one can seen in Figure 4.9, the chance to correctly guess the label

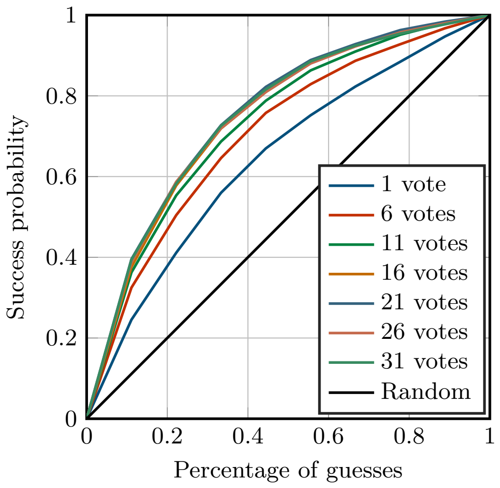


Figure 4.10: Not only the first guess accuracy improves, but all consecutive guesses are more accurate. More than 50% of gestures are classified correctly in two guesses.

in the first attempt increases with k. The chance dramatically increases for very small values of k. Performance stagnates around 40% when using more than 20 test gestures at once and starts deteriorating when using more than 35 test gestures. We believe that this is because of the limited training set size for users as discussed in Section 4.4.6. If we only consider users with more than 30 gestures per label, then these users need to have a lot of gestures and thus have played the game in varying environments. Hence, the more we increase k, the noisier the data gets. Thus, the advantage of having more votes to cancel out noise is balanced out by more noise introduced by later samples. As show in Figure 4.10 the performance not only increases for the first guess, but helps predicting the correct gesture with higher accuracy also for the following guesses. In our case, the peak performance of 40% of first guesses being correct was reached when using 27 test gestures. This means that an attacker can more easily guess pins or passwords that are repeatedly entered.

In addition to that, the attacker needs to solve the problem of recognizing repeated inputs. In case of device unlock pins or patterns, this is easily achieved through events triggered by the operating system.

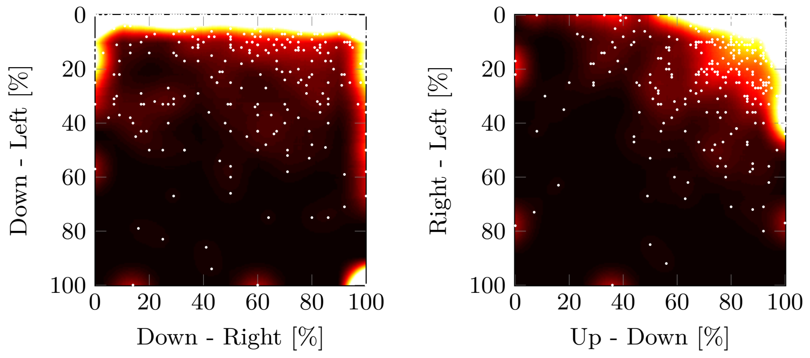


Figure 4.11: Gesture preferences of 452 users. “Down - Right” indicates a downward diagonal line from left to right, “Up - Right” an upward diagonal line from left to right. “Up - Down” is a downward, vertical line, “Left - Right” a horizontal line from left to right.

4.5 Additional Observations

We observed a heavy preference in which direction a gesture is performed. Most people have a preference on how they imitate a certain pattern. A pattern with a straight horizontal line can either be drawn with a swipe gesture from left to right or a swipe gesture from right to left. The same applies for vertical or diagonal lines. We analyzed the behavior of 452 users. The results are shown in Figure 4.11. One can see that most of the people prefer to perform the vertical gestures downwards and the horizontal gestures from left to right. When it comes to the diagonal gestures, there is an overall preference for the “Down - Right” gesture instead of the opposite direction, but for the “Up - Right” respectively “Down - Left” gestures, there is less of a general preference.

The profiles created in the previous subsection may reveal further information about the user. For single test persons, we observed that the gesture preference may be related to the handedness of the user.We could not confirm this claim, since we did not collect the corresponding ground truth data with the Android application. In future work, one could collect this data from test users to answer this question.

4.6 Conclusion

In this chapter, we discussed a side channel attack on touch input by analyzing motion sensor data. Firstly, we collected data by distributing an Android application. Secondly, we trained a *DTW* based classifier using the collected data to infer touch gestures. In contrast to similar work, we collected real world data in a way an attacker could also do. The evaluation has shown that the side channel attack presents a realistic threat. Especially for touch input which is repeated often, such as unlock patterns or pin codes, motion sensor data can help an attacker to guess the correct touch input.

As opposed to software vulnerabilities, the side channel attack we analyzed in this chapter is not caused and cannot be fixed by app developers. Background access to motion sensors needs to be limited on the operating system level because otherwise no application can protect itself against these attacks. Since we expect motion sensors to become more accurate in the future, the risk of a successful side channel attack grows even further. As devices are more capable in observing motion as well as their environment, the risk of side channel attacks will only increase.

5

Text in Motion

As we have seen in the previous chapter, the motion of a device can be used to infer what a smartphone user types on the screen. The introduction of smartwatches is putting accurate motion sensing devices around peoples wrists. Nowadays, these sensors allow to track the movement of a user's wrist in great detail. In this chapter we focus on the opportunities that these sensors present for novel user input methods. Existing touch input methods for smartwatches can be cumbersome due to small display sizes. The emerging options for voice input is efficient on such small devices. However it may be awkward to use in quiet environments or when private information has to be conveyed in public places.

In the following, we describe a system that can recognize letters by analyzing motion data recorded with a state of the art commercial smartwatch.[1] Generally speaking, we build a gesture recognition system and evaluate its performance on a set of 26 distinct gestures (the Roman alphabet). Letters are interesting since people do not have to artificially learn the gestures. However, also arbitrary gestures could be trained and tested. Rejecting- or accepting a phone call, muting an alarm, skipping or pausing a song are just a few of the commands that could be mapped to gestures. The similarity measure we use for this is based on Dynamic Time Warping (DTW). We

[1] We used a LG Watch R for our experiments, but any modern smartwatch can easily be used.

focus on creating a classification method which performs well with few training samples, which is why the system was tested using 3 training samples for each of the 26 letters. The classification method does not only provide the most likely gesture but creates a ranking of gestures depending on the similarity to the input. Therefore, we could also predict the second likeliest gesture and so on. We also apply our method to written words. However, naively segmenting the data has proven infeasible. We show how the pen's sound can be used to extract isolated strokes, which greatly simplifies the segmentation problem.

5.1 Related Work

Gesture driven interaction is an extensively studied problem. The methods employed vary from pattern recognition in camera images [36], using special gloves to record the data [47], to more similar technologies to the smartwatches used in this paper, e.g., a Wiimote [72]. Consequently, smartwatches themselves have also been used for gesture recognition, e.g., in [12, 68, 69]. In contrast to these works, our system can cope with gentle gestures that only cause minor movements of the smartwatch.

A more specialized version of gesture recognition is handwriting recognition. The most prominent solution is optical character recognition (OCR). Instead of trying to analyze the movement, the resulting image is being looked at. This process does not use any time information and is similar to how humans do it. It has a long and successful history [67]; using various techniques like hidden markov models [21] or SVMs [7].

We want to focus on approaches that use the movement of the hand. There is software to recognize written text on touchscreens [32, 44]. But since the screen is two dimensional and provides very accurate information about the x and y coordinate, techniques used in this line of work are hard to translate into our setting. Text recognition on whiteboards has been explored using infrared [50]. This approach is less easily applied than our method in everyday situations because it requires infrared transmitters and receivers attached to the whiteboard.

Closer to our system is research from Choi et al. [27]. They use an accelerometer directly attached to a pen to recognize written numbers. This system needs a special pen; it is evaluated using the 10 Arabic numerals, and achieves a recognition rate of more than 90%. Even though the smartwatch in our work may be seen as special hardware, it could be used for much more tasks than just recording handwriting and might become, like smartphones, an ubiquitous companion in the future.

In a recently published work Xu et. al. have also used a smartwatch-like device carried on the wrist, to realize gesture recognition [85]. They also implemented a letter recognition system which recognized up to 94.6% of the letters on average. They restricted the letters to be within a window of fixed size, while the arm has to lay on the armrest of a chair. The letters were drawn using the index finger. With our system the arm position does not have to be fixed, and the letters can be written with a normal pen.

Smartwatches also have been used in an unexpected way to predict words that are typed on a keyboard [83]. The smartwatch is attached to the left wrist and depending on the movement of the wrist, a list of words that is likely to contain the written word is generated. A median of 24 words is sufficient to correctly guess the typed word. This number decreases for words that are longer than 6 letters to 10.

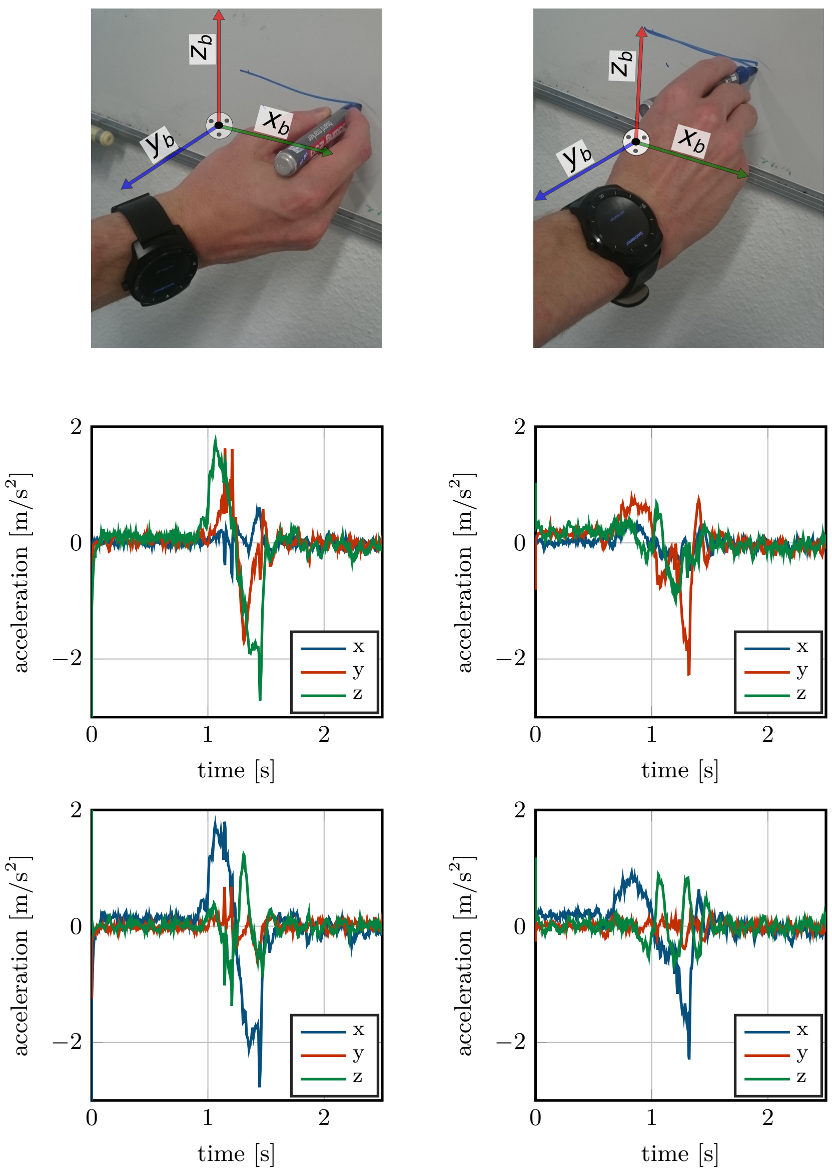


Figure 5.1: Drawing a straight line parallel to the x_b axis leads to different measurements depending on the watch orientation. The left column shows the linear acceleration measurements when the clock face points to the right. The right column corresponds to the measurements where the clock face points upwards.

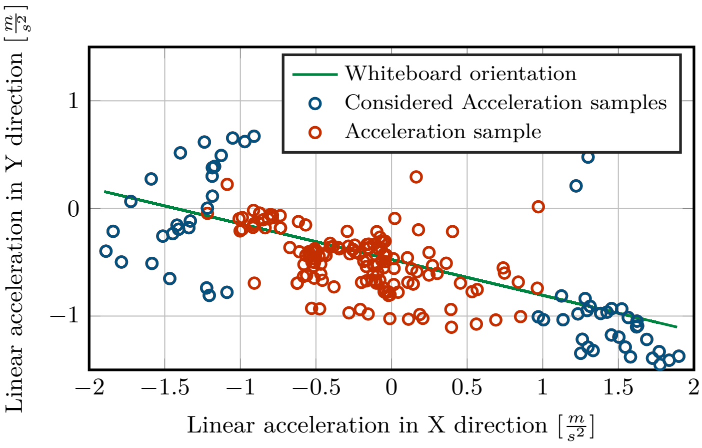


Figure 5.2: The variance of the linear acceleration is large with respect to both axes. The green line represents the orientation of the whiteboard. Only the blue points where considered to fit the blue line

5.2 Method

The raw sensor data is influenced by changing hand postures as shown in Figure 5.1. Furthermore, writing at the top or bottom of the whiteboard affects these measurements. Training our system to work in all these environments is cumbersome. This would artificially increase the number of gestures we have to recognize which in general reduces the recognition performance.

To alleviate this problem, we transform raw measurements in sensor coordinates to a coordinate system relative to the whiteboard. This allows us to track user motion with respect to the whiteboard. The whiteboard coordinate system is invariant to changes in watch orientation relative to the user's wrist. Hence, we can reuse training data even if the watch is not worn exactly the same when performing the gesture. First we transform the measurements into the world coordinate system to remove the acceleration caused by the gravity of the earth. The world coordinate system consists of the z_w axis pointing to the sky, and the x_w and y_w axes lying perpendicular in the horizontal plane. No magnetometer measurements are used to relate x_w or y_w to the cardinal directions. Instead, we rely on a Kalman filter [43]

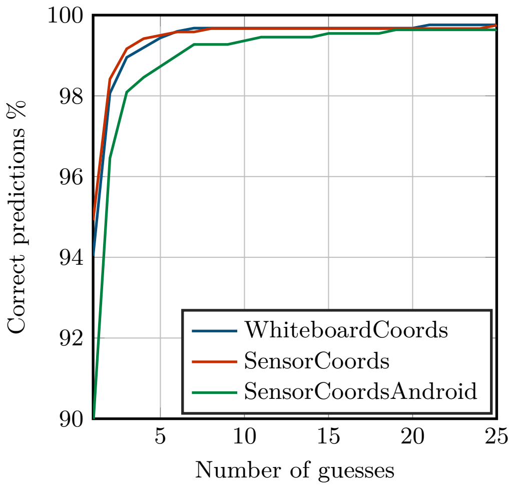


Figure 5.3: The recognition performance for each feature. The plot shows how many letters were correctly recognized depending on the number of guesses available. After 3 guesses WhiteboardCoords correctly recognizes 99% of the letters. Even though this is slightly worse than SensorCoords, this feature is superior since it can also handle letters written on a different height (see Figure 5.4).

to determine the z_w axis.

The z_w axis already directly relates to the whiteboard z coordinate z_b. To find x_b and y_b we assume that writing on a whiteboard mostly causes acceleration within the whiteboard plane. As a result, the x_w and y_w coordinates of the linear acceleration measurements (gravity compensated) are correlated. We use large accelerations to find the dependency caused by the whiteboard through linear regression. Figure 5.2 shows this correlation along with the determined whiteboard orientation. As a result we can find and track the base vectors of the whiteboard coordinate system. Since this is ambiguous, we use both possible rotations in the recognition process (the y_b axis pointing into the whiteboard and pointing out of it).

Android provides functions to transform measurements in device coordinates into world coordinates. As our results show, this transformation does not work reliably. We want to avoid to use the magnetic field sensors since errors can be introduced by magnetic fields caused by currents, magnets or magnetized material.

The recorded training gestures also contain irrelevant data before the gesture starts and after it ends. Since during training, the users are required

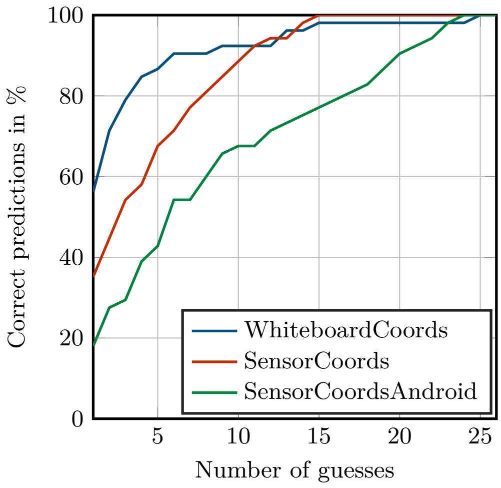


Figure 5.4: The effect of transforming the data into whiteboard coordinates. The test gestures were written at the upper edge while the train gestures were written at the lower edge of the board. The transformation improves the result.

to keep their hand still before and after writing, we use the amplitude to segment our data.

The data is recorded during a short time interval. The time from the beginning of this window up to the moment when the user starts to write is called *left-margin*. Similarly, the time before the user finishes writing and after the end of the recording period is called *right-margin*. These margins should contain no movement of the pen. The measurements which correspond to the gesture we are interested in, lay between these two margins. To eliminate these margins and extract only the relevant part of the gesture, a segmentation algorithm is used. The algorithm relies on changes in amplitude of the acceleration signal to find the start and the end point of the gesture. The amplitude corresponds to the euclidean norm of each sample of the sequence. If the amplitude is larger than a given threshold, we assume that there was pen movement. The first time the amplitude is higher than the threshold will be interpreted as the starting point p_s. If there is a longer period of time, after detecting such a p_s, in which the amplitude is continuously lower than the threshold, this will be interpreted as the end of the gesture and an endpoint p_e will be set. Extracting the sequence of measurements between p_s and p_e eliminates the margins. In addition, we want to eliminate noise, which is induced by the user before writing a letter, e.g., if he moved his hand before the application told him to do so. To achieve this, the algorithm does not stop after finding the first p_s, p_e pair, but it also considers the rest of the sequence. If there is a new value larger than the threshold, the algorithm rejects the already collected start-/endpoints and chooses the new interval. This removes noise in the beginning, but may still produce wrong segmentation values if there is noise in the end. Note that the collected data seems to indicate that noise appears mainly in the beginning.

5.2.1 Classification

The classification algorithm for letters consists of three stages. The first stage extracts features from the collected data. We evaluated various features of which the following performed best:

- **WhiteboardCoords** Gyro & linear acceleration data transformed into the whiteboard system as described above.
- **SensorCoords** Linear acceleration & gyro data in sensor coordinates.
- **SensorCoordsAndroid** Linear acceleration & gyro data provided by Android in sensor coordinates.

In the second stage, the sequences of feature vectors are compared using the DTW algorithm [59]. To compare two feature vectors, we use the euclidean distance.

In the third stage we create a ranking out of the similarity scores between test- and training-letters. This ranking captures the similarity between the input and all the training samples of all letters.

5.3 Evaluation

Recall that the recorded data depends on the orientation of the watch because the device may change its orientation depending on the height the user writes on the board. Therefore, we implemented the transformation into the whiteboard coordinate system as described earlier in order to obtain orientation independent acceleration measurements.

Figure 5.4 shows the difference between non-transformed and transformed data. In this case the whole alphabet was written by one of our users above the lower edge of the whiteboard and at the upper edge twice. The gestures written at the top are compared to the gestures written at the bottom and vice versa. We used the features WhiteboardCoords, SensorCoords, and SensorCoordsAndroid. We test two gestures of each letter

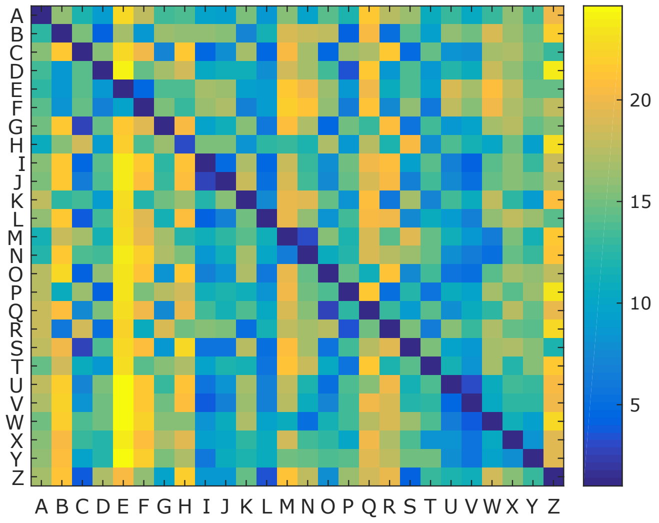


Figure 5.5: This matrix represents the average ranking for each letter and all users if we use the WhiteboardCoords feature. The test letters are shown on the Y-Axis, the values on the X-Axis represent the ranking.

against two training gestures for each letter. As we can see, we get the lowest performance using the linear acceleration provided by Android. In the remainder, both the test and the training set were written on approximately the same height.

Twelve people were asked to write each letter of the alphabet four times. The test subjects were asked to write at a comfortable height without further restrictions. We test the algorithm offline by using three samples of each gesture as training set and one for testing with cross validation.

5.3.1 Letter Recognition

In Figure 5.3 the performance for all users is shown. Using our rotation into whiteboard coordinates slightly reduces the performance to about 94%, but it allows us to use the training data on different whiteboards. Within three guesses, we can predict the correct letter in 99% of the cases. The percentage of letters that was predicted correctly after a specific number of guesses is shown for each user. This shows that the rotation into whiteboard coordinates clearly outperforms the linear acceleration provided by Android.

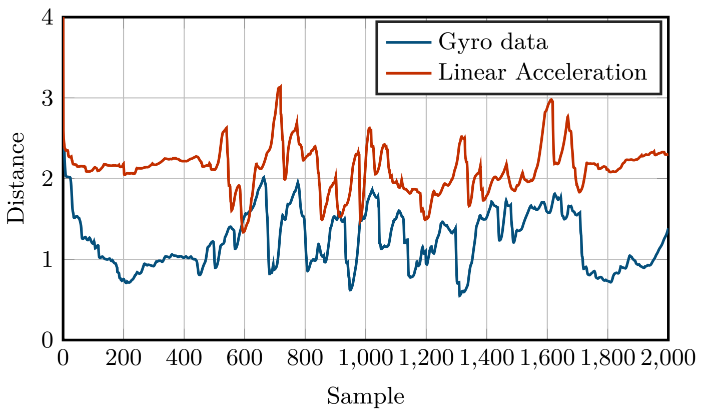


Figure 5.6: Distance for each feature of the first guess of every evaluated point in time when using a naive approach on the word "JUMPS". The red and green marks show the start and end points of the letter according to the ground truth.

5.3.2 Letter Confusion

Figure 5.5 shows the average similarity rank for each test letter (Y-Axis). The diagonal values have the best (lowest) entries. This means that our system predicts low ranks for the correct letters on average. The entries for the four letter pairs ("I","J"), ("M","N"), ("E","F") and ("U","V") shows that these pairs are easily confused. This makes sense, since these letters look somewhat similar. Generally, the letter "E" can be distinguished well. This can be explained by the relatively long time and the number of strokes it takes to write an "E". Note that this matrix does not need to be symmetric. We take the column of the letter "E" as an example. "E" appears late in the ranking for many other letters but this does not mean that all letters appear late in the ranking of an "E". Many of these letters will produce high costs comparing them to an "E".

5.3.3 From Letters to Words

In order to extract letters from unsegmented sensor recordings, we need to find starting points of gestures. If the user writes quickly, then segmenting the individual letters is non-trivial.

First, we naively assume that each sample in the recordings is a starting point and generate a ranking of letters for it. However, finding the starting points of gestures from this sequence of rankings has proven to be

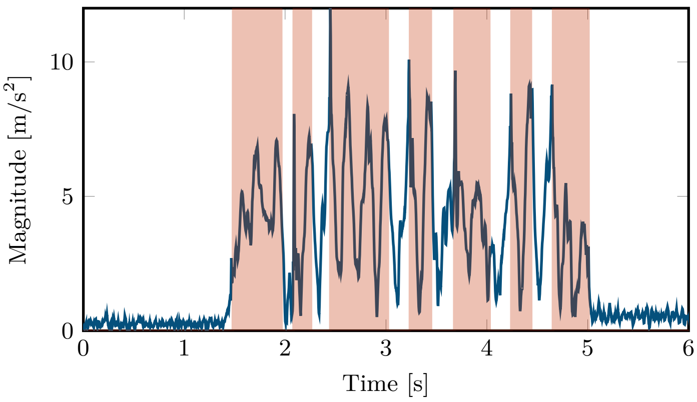


Figure 5.7: Cross correlation between an audio signal and the corresponding acceleration measurements. Red areas indicate strokes as detected and in Figure 5.8.

difficult. Figure 5.6 shows that the *DTW* distances do not reliably drop significantly when a gesture starts. Hence, this method does not work directly. Furthermore, for most (random) time segments, there is a trained letter that matches the sensor measurements in said segment. This means that in order to find as many correct gestures as possible, we have to accept a massive number of false positives caused by the difficulty to segment the input correctly.

To reduce the number of false positives, we segment the sensor measurements using audio recorded with the smartwatch while writing. When a pen is dragged on the whiteboard surface, it emits a distinct sound that can be used to track isolated strokes on the whiteboard.

Figure 5.8 shows the spectrogram of an audio track recorded when writing “OVER”. The strokes of each letter are easily visible. Our audio assisted recognition algorithm uses this information and only applies our *DTW* distance metric whenever a stroke starts. We call the starting points of strokes *Points of Interest* abbreviated as POI.

The detection algorithm sums up the absolute amplitude value of all frequencies between 5 kHz to 10 kHz for each point in time. This gives us a vector $\mathbf{p}$ where $\mathbf{p}_i$ contains the sum for the i^{th} point in time. We then search for values in $\mathbf{p}$ which are greater than a threshold value τ. Multiple successive values greater than τ will be combined.

Note that the recording of the audio and the recording of the acceleration do not start exactly at the same time. Because of that, we align the two signals using the cross correlation between $\mathbf{p}$ and the evolution of the

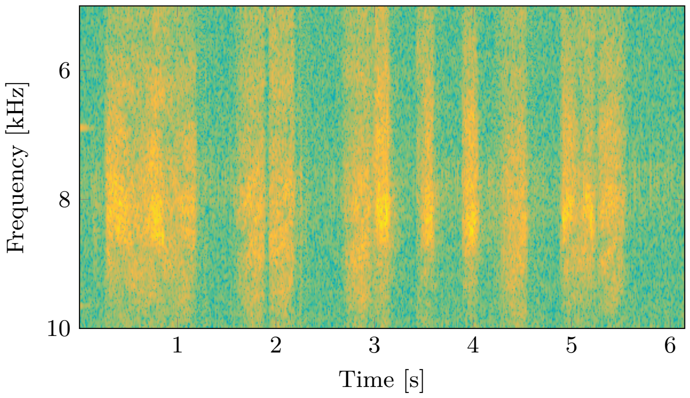


Figure 5.8: Spectrogram of the word "OVER". The strokes are clearly visible for each letter. "O" shows as one long stroke ending shortly after the first second. "V" is split into two strokes at two seconds. "E" starts at three seconds and ends at second four and "R" contains two strokes, the latter of which shows the change in direction in the same manner as "V".

magnitude of the linear acceleration. An example is shown in Figure 5.7. Now, we are able to extract the starting and ending of a stroke with respect to the linear acceleration signal.

We run our *DTW* distance metric for every POI. Because letters may consist of multiple strokes, we cannot use the next stroke endpoint to extract the test sequence. Instead, we use a time window of length 2.5 seconds to select possible endpoints. As a result, every starting point leads to multiple test sequences, which are individually passed to the recognition stage. In total, eight users wrote 428 words. Half of these words were written with a small break between the letters; the other half without any restrictions. The ground truth was manually annotated. We evaluated the performance using the WhiteboardCoords feature, which performed very well in the letter recognition task and also works for arbitrary writing heights.

The naive approach, considering each sensor sample as a possible starting point for a letter, correctly recognized 64.5% of the letters within 3 guesses. However, the method produces nearly 250000 erroneous predictions. Most of them in places no letter started.

Using the audio signal to reduce the number of possible starting points of gestures increased the recognition rate to 71.2% within three guesses. It also allowed us to reduce the number of erroneous predictions by nearly two orders of magnitude.

5.4 Conclusion

Our experiments show that recognizing which letters are written to a whiteboard using motion data recorded with a smartwatch is feasible. This holds true even if the training set is small. Hence, our system could be used to recognize gestures, e.g., to cycle through slides while writing on a whiteboard. Recognizing letters within words is possible to some extent if we know when the letters appear, i.e., the start and end time. However, without solving the segmentation problem, our approaches lead to prohibitively large numbers of erroneous recognitions. Using the audio data helps reducing the number of false positives but the performance is still not good enough for practical applications. Similar to software keyboards or T9, we may be able to estimate the most likely word even based on ambiguous letter input. Digressing from the task of recognizing text, the recognition of isolated letters and hence gestures performed well. Even with 26 different gestures that make the classification task harder, more than 94% of gestures could be correctly recognized.

6 Discussion Diarisation

Modern technology has allowed us to quantify more and more aspects of our behavior. A few years ago, fitness tracking for example was limited to comparing lap times, running distances or other key parameters of a workout. Today, fitness tracking devices can tell you, with high accuracy, how much exercise you get throughout a normal day by continuously monitoring your behavior. The ease of quantifying complex personal exercise behavior leaves a user with undeniable facts, that when taken seriously, can help improving quality of life. The same principles have been applied to social interactions. For example, there is a tool that helps quantifying the quality of the encounters with your friends [77].

Face to face communication is a vital part of our everyday lives. Discussions occur at work or with friends and family. However, even though we spend a lot of time talking to other people, it is hard to obtain objective data about these discussions. The lack of facts, be it during business meetings or in informal situations, makes it hard to improve discussions. Also, we cannot present our peers with facts when criticizing or trying to improve a conversation. Subjective criticism can be perceived as being offensive rather than helpful. In a corporate environment, inefficient communication and a bad work climate directly translate to added cost. To reduce these effects, companies organize team building events or even hire counselors.

In this chapter we evaluate how discussions can be analyzed in an unobtrusive way. Like in the previous ones, we use measurements performed with regular smartphones. To our knowledge there are no tools to easily track verbal interactions, even though conversations are an integral part of our everyday lives and the underlying mechanics might give insights on the state of a relationship. The information can be used to identify behavior that is keeping the discussion from being productive. For example, there might be a person hogging the conversation by not leaving any room for others. Or there might be someone who continuously interrupts others. Telling the culprit is usually difficult since there is no evidence and hence, the constructive criticism may be ignored or interpreted as a personal attack. By supplying objective data of such behavior, conversations can be optimized in an objective way.

The tool we developed to this end (*RTDS*) can accurately capture how people interact in a discussion. It can give insights about how much people speak, or reveal more complex patterns in their interaction. *RTDS* is applicable in many situations such as business meetings, interviews, dinner conversations, or during arguments. Running on of the shelf smartphones, *RTDS* is quickly setup and can give insights about verbal interaction patterns wherever you go. Furthermore, *RTDS* can also augment conversations by acting as a referee, e.g., by notifying participants of using too much time in the ongoing conversation.

6.1 Related Work

Business meetings have been in the spotlight for being inefficient and frustrating as shown in a study by Romano et al. [71]. The process of automatically distinguishing different speakers is called speaker diarisation and is extensively discussed in literature.

One of the prevalent types of systems rely on acoustic features like Mel Frequency Cepstral Coefficients (MFCC) [60] and others [58] generated from one recording to identify the active speaker. These systems are especially useful if only one recording is available such as during radio broadcasts. However, we have observed that MFCC features perform poorly for voices that are similar (such as the voices of brothers). MFCC features are suitable for authentication tasks when the spoken words are always the same. Changing the content introduces uncertainty that greatly reduces the performance of these features.

Our discussion mechanics inference method is based on *Time Difference of Arrival (TDoA)* to determine the *Angle of Arrival (AOA)* of sound signals. Humans and animals use this method in everyday life with their two ears to aid the localization of sounds, we refer to [73] for an overview: As it turns out, the change of the angle of a sound source can be detected by humans even if the difference is as small as 1°, depending on the relative position of the sound source. Not surprisingly, this established technique has already been applied in various contexts, e.g., shooter detection: In Washington, D.C., USA, the installment of just 300 sensors was enough to localize several ten thousands of gun shots in an area of 20 square miles since 2006 [66].

More directly related to the work in this chapter are methods that are using TDoA to determine the relative speaker location/angle (cf. [19, 38]) or to localize a set of connected smart phones in a meeting, cf. [65], where 10 phones are used for accuracy. However, all these systems differ from the work in this chapter in the sense that we just need one off the shelf smart phone with our application installed. No specialized hardware or cumbersome setup is needed. The participants enroll with one tap on the screen.

McCowan et al. [55] presented a system that automatically analyzes and stores meeting contents. Amongst others, features such as "speech pitch" and "presentation speech activity" are tracked. The features allow for fine grained analysis of the meeting. However, the system requires multiple cameras and microphones being placed in a certain way. This reduces the applicability in many real world scenarios in which *RTDS* can give insights on the social dynamics of a discussion.

Another aspect of our work relates to *lifelogging* (cf. [9,22]) & *the quantified self* (e.g., [79]) and augmenting conversations with smart phones. Many people are interested in quantifying activities in their everyday life by logging (for example, recording audio with their smart phone all the time [74]) and subsequently analyzing them. Lu et al [51] recently discussed continuous audio sensing to identify nearby speakers could improve life-logging applications. They use a single microphone to determine if a certain speaker is talking at the time. Note that their approach requires training for each speaker that is to be classified whereas our method does not require any training data. Similarly, Xu et al. [86] showed that smartphone microphones can be used to count speakers in an unsupervised fashion.

Automatically augmenting discussions can come into play here, by, e.g., trying to enforce certain conversation criteria (as done in our work). Current applications such as the system in [41] embark in a similar (yet maybe orthogonal) direction by creating, e.g., tickets-to-talk between participants.

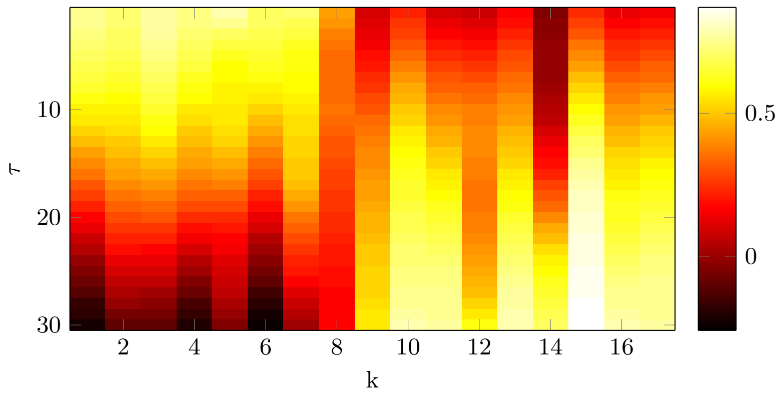


Figure 6.1: Values of $c(\tau)$ for different time slices k (44100 samples each) and sensible values of τ considering a sampling rate of 44.1kHz and microphone distance of 12.7 cm. At $k = 8$, the TDoA changed drastically which is clearly reflected in the change of $c(\tau)$.

6.2 Measurement System

In order to distinguish people within one discussion, we rely on a Time Difference of Arrival measurement of the audio signal. We use a Samsung I9300 smart phone from 2012 which allows for stereo recording using the two built in microphones. A large portion of current smart phones is equipped with two microphones, with some even having three or more (e.g., LG G3). The microphones are sampled at 44.1 kHz and are located at the top and bottom of the device (12.7 cm apart). Assuming the spoken signal is $s(t)$, we assume that we receive $r_i(t) = k_i \cdot s(t - \tau_i) + n(t)$ at microphone i. The noise term $n(t)$ is assumed to be uncorrelated and the signal $s(t)$ is shifted in time due to the distance between microphone and the sound source. The anisotropic gain of the microphones and the speaker itself is captured in k_i as it is constant during a discussion as long as we assume neither microphones or participants move significantly. To find the angle of arrival based on two microphones ($i \in [1, 2]$), we use a cross correlation of the received signals: $c(\tau) = \sum_{i=0}^{n} r_1(i) \cdot r_2(i - \tau)$.

This gives us an indication of how well the received signals match for a given time delay τ within a fixed time slice of length n samples. The value τ which maximizes $c(\tau)$ gives the time difference of arrival at which the most energy arrives at the two microphones. Since each τ corresponds

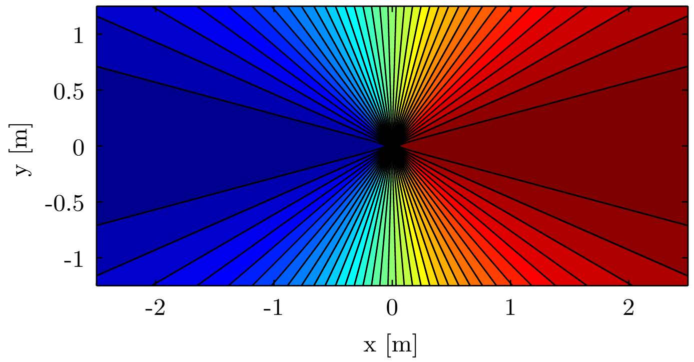
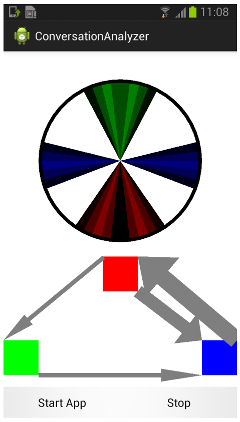


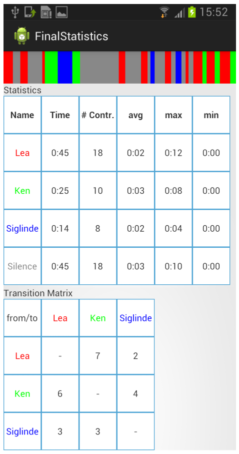
Figure 6.2: Areas of same color lead to the same TDoA measurement at the phone. The area shown is 5m by 2.5m and the microphone assumptions are based on the Samsung I9300 which we used for our experiments. As seen, TDoA with two microphones cannot distinguish two speakers sitting on opposite sides of the microphone axis.

to an angle of arrival given the distance between the microphones (shown in Figure 6.2, the speakers can approximately be localized with respect to the microphone pair. Figure 6.1 shows values of $c(\tau)$ for different delays τ over a period of 17 consecutive time slices of one second. The change of $c(\tau)$ at half time of the measurement indicates that the angle of arrival has changed significantly. See Figure 6.2 to get an intuitive idea of which areas around the microphone pair lead to a fixed value of τ. As you can see, the areas extend to both sides of the microphone axis. Therefore, two speakers sitting exactly opposite of each other with respect to the microphone axis cannot be distinguished using our method and only two microphones. However, this problem can easily be avoided by placing the phone such that the arrival angles are unambiguous. Should there be no dominant angle of arrival ($c(\tau) < x \; \forall \; \tau$), we assume that nobody was speaking. The threshold we use to separate silence from speech is empirically evaluated (we refer to the evaluation section for details).

Discussion Dynamics Inference. In order to reduce the effect of measurement noise and to infer high level discussion statistics, we use a Markov model. The structure of the model is defined as shown in Figure 6.4. Each user can either be speaking or silent (top and bottom row of states). By modeling silent states for each user, we allow the model to distinguish a pause from a user continuing the discussion after another stopped talking.



(a) Information shown by *RTDS* during a conversation. The circle shows the arrival angles for each user in a different color. The symmetry of these arrival angles is caused by the fact, that two microphones only allow for unambiguous angle arrival detection to either side of the microphone axis. Below, the transition probabilities between all the speakers in the discussion are shown. Wider arrows indicate higher transition probabilities.



Statistics

Name	Time	# Contr.	avg	max	min
Lea	0:45	18	0:02	0:12	0:00
Ken	0:25	10	0:03	0:08	0:00
Siglinde	0:14	8	0:02	0:04	0:00
Silence	0:45	18	0:03	0:10	0:00

Transition Matrix

from/to	Lea	Ken	Siglinde
Lea	-	7	2
Ken	6	-	4
Siglinde	3	3	-

(b) An example for the final statistics shown by *RTDS*. Included is the speaking time for each participant (Time) and the number of distinct contributions (# Contr.). In the lower part of the screen the complete transition matrix between the users is shown (silent and speaking states for each user are fused). The bar at the top displays who spoke during which time period.

Figure 6.3: Two of the display modes of *RTDS*.

The transition probabilities are adapted throughout each discussion to best explain the sequence of angle of arrival measurements using the Baum-Welch algorithm, cf. [52]. The emission probability distributions for each speaking state matches the dimensions of the TDoA measurements. Figure 6.6 shows such a distribution that resulted from one of our test discussions. Each silent state is only allowed to emit silence observations. This means that our system does not require prior information about where speakers are located, and is able to find and distinguish speakers solely based on the structure of the Hidden Markov Model. To generate the final statistics, we use the Viterbi algorithm [82] to obtain the most likely state sequence using the trained model.

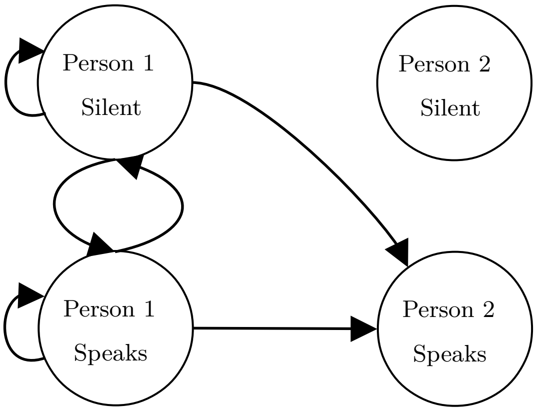


Figure 6.4: Hidden Markov Model template for two speakers. Silent states are introduced for each user to capture user specific discussion behavior. For simplicity, only transitions originating from Person 1 are drawn.

6.3 Application

We implemented the discussion inference method described above in an Android application. The application requires two microphones that can be programmatically accessed which most modern phones provide. In order to obtain statistics for a discussion, each participant needs to specify an alias and an approximate direction so the states of the Markov chain can be matched to the user aliases.

During any discussion, *RTDS* provides on-line feedback about the current state of the discussion as shown in Figure 6.3a.

Offline Feedback. In addition to the online feedback, *RTDS* provides detailed statistics after each discussion, cf. Subfigure 6.3b. These include the total amount of time each participant spoke and how many distinct contributions were made. Also the min, max and average length of each contribution is listed for each participant. The transition matrix shows how many contributions from participant A were followed by a contribution of participant B. This information can be useful for finding repeating patterns in the way the participants interact.

Special Operation Modes. In addition to the passive feedback modes described above, we can use the online statistical data to enforce constraints on how participants may interact. In case a constraint is violated, we can give visual or acoustic feedback. We included two operating modes

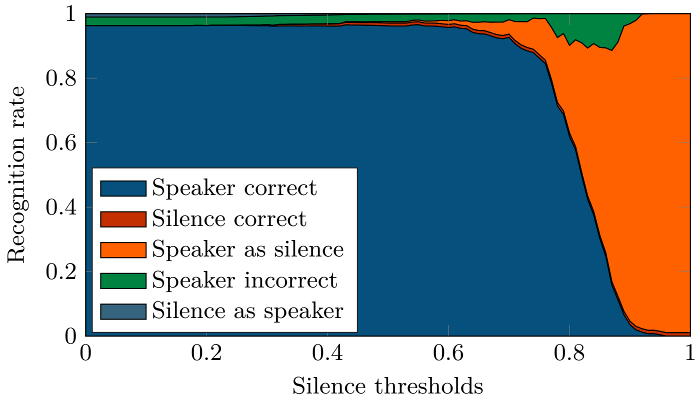


Figure 6.5: The graph in this figure depicts how the silence threshold (in $\frac{\max(c(\tau))}{\text{sum}(c(\tau))}$) affects the classification accuracies of both speaking and silent states.

into *RTDS* that the user can choose from. The first mode helps enforce even speaking times for all participants. The second mode forces participants to take turns when talking and limiting the talking time for each participant and round. These modes for example can be used to defuse a heated argument by disallowing repeated interruptions or by reproving participants that are not allowing others to talk.

6.4 Evaluation

To evaluate the accuracy of our system, we recorded 6 separate discussions with a total of 7 people. Each of the 6 discussions was annotated manually, this information was then used as ground truth. In all 6 discussions, two (1), three (3), or four (2) people were involved, and the phone was placed in a way that allowed for unambiguous arrival angles for all participants. Natural disturbances like coffee machines or other background noise were present. Figure 6.5 shows the overall classification results with respect to a varying silence threshold. We consider direct transitions between speaking states of two users a valuable metric that captures how one user might disrupt another. Therefore, we consider confusing speakers worse than confusing a speaking state with silence since the latter does not affect direct transition probabilities between speaking states. With a growing number of observations being classified as silence, more speaking states are confused with silence. Most wrong speaking user classifications occur at a silence

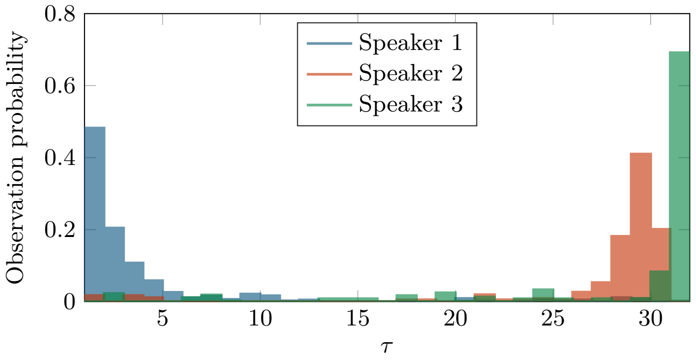


Figure 6.6: The observation probability distributions are accurately estimated even if two users cause similar time of arrival differences τ depicted on the x-axis. In this case, speaker 1 and speaker 2 are only separated by a two sample delay between the microphone channels.

threshold of 0.8-0.9. For lower threshold, the model matches the discussion better thereby causing less errors. For higher thresholds, too many speaking states are mistaken for silence, thereby reducing the share of speaker misclassification. According to our results, a silence threshold of 0.5 is ideal as it leads to error rates of 3% and erroneous speaking user classification rates of only 2%.

All our experiments were carried out with up to four participants. However, the number of speakers is not limited to four. Figure 6.6 shows the observation probabilities in a discussion during which two speakers (2 and 3) sat very close to each other and therefore caused similar time difference of arrival measurements. Even though the measurements are only 2 samples apart on average ($45 \mu s$), speakers are reliably detected and misclassification rates are only 2%.

6.5 Conclusion

In this chapter we looked at speaker activity annotation using one single smartphone with a stereo microphone. The presented application collects speaker activity data online and aggregates statistics which are then presented in a summary. Furthermore, *RTDS* can be used as an active referee to enhance discussions by, e.g., allocating a fair share of time to each participant. For an early prototype version of *RTDS*, we experimented with voice recognition to identify participants: While the recognition rates were in general still acceptable (mostly close to Angle-of-Arrival), there were accuracy issues with recognizing very similar voices, e.g., brothers. In the next chapter, we show how speaker ambiguities can be resolved when multiple smartphones are used to collaboratively annotate speaker activity.

7

Distributed Discussion Diarisation

Pursuing the same goal as in the previous chapter, we extend our discussion diarisation system to aggregate data from multiple smartphones. This helps resolving ambiguities caused by stereo microphones but requires overcoming severe hardware limitations of current smartphones. Since most people carry a smartphone nowadays, *Disca* can be applied in most everyday situations easily. We collect this data using a set of smartphones that collaborate to identify the current speaker. Each smartphone records the conversation and exchanges chunks of recorded audio with the others. For each smartphone pair, the delay between the recordings is estimated using cross correlation. This leads to a vector containing delays for each smartphone pair which is then used to identify a speaker. Our method compensates offsets in the sampling rates of different smartphones and runs in real time on off-the shelf smartphones. As in the previous chapter, a Markov model is used to reduce the effect of noisy measurements. The computational burden is distributed among the participating smartphones to avoid very slow devices being overburdened. The results are visualized in real time and archived so previous conversations can be aggregated or compared. Also, the results gained from the Markov chain allow to analyze if there are cliques of participants communicating mostly with each other.

Disca performs all computations in real time without sending any audio recordings to the cloud. Instead, all computations are performed locally such that no personal data has to be shared to obtain the results. To our knowledge there are no speaker diarisation systems that can run in a distributed setting. This is mostly due to clock inaccuracies that prohibit tracking time difference of arrival measurements. In *Disca*, we show how clock inaccuracies can be overcome by coarsely synchronizing the clocks via network as well as tracking clock drifts using the recorded audio directly.

7.1 Related Work

We refer to the last chapter for related work focused on discussion diarization. In this chapter, we focus on systems takes advantage of multiple microphones. In contrast to methods relying on acoustic features, these methods generally require the microphones and speakers to remain approximately in the same location during a discussion, the speaker voices can be arbitrarily similar. Brandstein and Silverman [19] showed that microphone arrays can track active speakers. Similarly, Anguera et al. [3] use acoustic beamforming to enhance the signal from multiple distant microphones. However, the time difference of arrival (TDoA) data is not used to classify speakers. In their later paper [2], recordings from different microphones are compared to a reference and the timing data is used to infer active speakers. Note that they need reference microphone which can record each speaker well. Hence, all speakers have to be at a similar distance to the reference microphone. If this is not the case, the results will deteriorate since it will affect all the TDoA measurements from all other microphones. In addition to this, all the above methods are not robust against uneven sampling rates across different recording devices. We show that off-the-shelf smartphones are equipped with clocks that are prohibitively inaccurate for the above methods to work. More recently, Sur et al. [78] showed that smartphones can be accurately synchronized to perform beamforming. A central server is used to track clock drifts to achieve array gain for the microphones. Their speaker localization algorithms require the phones to be placed according to a given scheme. Also, a central server is required to achieve accurate synchronization. *Disca* does not require a central server or reference microphone and can accurately compensate for clock differences between recording devices.

Praviainen et al. [65] show how environmental sounds can be utilized to synchronize and localize off-the-shelf devices such as smartphones. Our system is similar because the recording setup of multiple smartphones is alike. Interestingly in [65], clock drifts are neither handled nor mentioned albeit in our experiments their impact on performance proved to be severe. Generally, the resulting sequence of clusters that best match the observations are post processed to reduce noise. For example, the Viterbi algorithm can

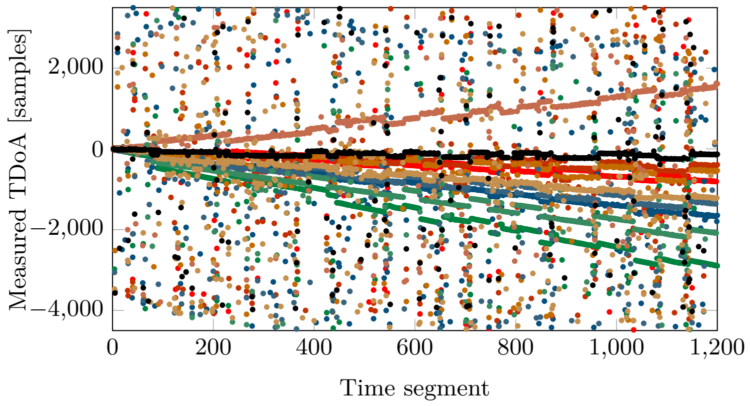


Figure 7.1: Typical drift observed between all pairs of five different phones. Each time segment corresponds to 8192 samples and hence 1200 time segments correspond to 220 seconds. The clock drifts exceed the time difference of arrival measurements by far after a short period of time.

be used to impose basic temporal properties of discussions as described by Anguera et al. [2].

7.2 Method

We aim to analyze discussions based on who was speaking at what time. To this end, a set of smartphones record the discussion. We assume that any two participants of the discussion have a unique set of distances to each microphone. If there are three microphones that are not arranged on a line, it is easy to see that, in a plane, there are no two locations with the same set of distances. Not every participant requires to provide a phone to obtain accurate results. The distances between the speaking participant and the microphones cause a propagation delay. If the locations of the microphones were known and their clocks would be synchronized, it would be possible to deduce the location of the speaker. Mostly because of the delay caused by the operating system which is not designed for such tasks, clock synchronization on such a high level of accuracy is infeasible on current smartphones.

We use the differences in propagation delays Δ_s to classify each speaker s. This difference is defined and may vary for each pair of smartphones. We assume speakers and smartphones to remain more or less in the same location throughout the discussion. In this case Δ_s is constant for all speakers

s.

The time difference of arrival (TDoA) $d_k(i)$ observed in audio segment i for the k^{th} pair of smartphones is influenced by the difference in sampling rates of the two recording smartphones r_k. Also, clocks are not perfectly synchronized which leads to a constant offset c_k. The time difference of arrival observation $d_k(i)$ therefore relates to the difference in propagation delays $\Delta_{s,k}$ as follows:

$$d_k(i) = \Delta_{s,k} + i \cdot r_k + c_k + w \tag{7.1}$$

We account for Gaussian measurement noise with the term w. When n smartphones are used, the time differences of arrival $d_{i,k}$ are calculated for each of the $n(n-1)/2$ pairs of recordings in each audio segment. Combining all pairs of recordings we get the following:

$$D_i = \Delta_s + t \cdot R + C + W \tag{7.2}$$

In the following sections, we show how we estimate the difference in propagation delay for each observed audio segment. This information is then used to find each speaker's Δ_s. First, the smartphones are roughly synchronized such that audio segments from the individual smartphones that were recorded roughly at the same time can be compared. The time differences of arrival D_i are then calculated for each audio segment as described in the Time Difference of Arrival Section. The estimation of the difference in sampling rates R is explained in the Clock Sync Section. The resulting vectors of propagation delay differences Δ_s are then estimated by clustering and filtered as described in the Clustering Section.

Figure 7.1 shows the raw TDaA measurements performed for a set of five phones. Each pair of phones leads to a line that is sloped because of the clock differences r_k between the two participating devices. Also, the influence of the audio source Δ_s is apparent since the lines for each phone pair assume different levels as the speakers take turn.

7.2.1 Time Difference of Arrival

The calculation of the TDoAs requires the recordings of the different smartphones to be roughly synchronized. To find the corresponding position of one audio segment in an other recording, the time difference should be small. Otherwise the audio segment has to be compared to a long segment of the second recording.

Before starting with the recording, the phones exchange packets analogously to the Precision Time Protocol (PTP).

Using this synchronization method, the smartphones start recording at roughly the same time. The audio is then partitioned into segments of 8192

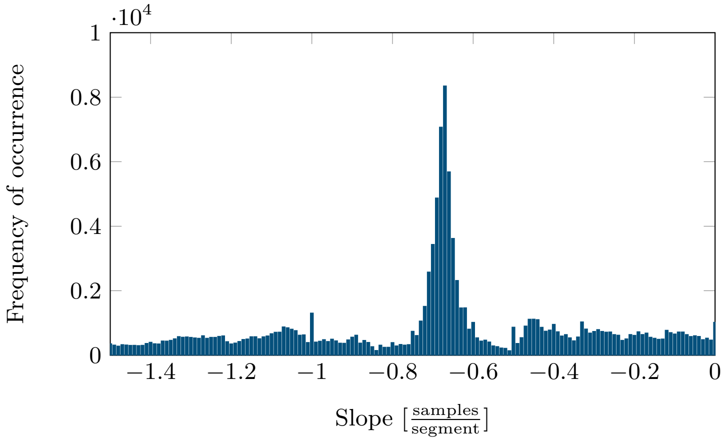


Figure 7.2: Typical voting result for the most frequent slope aggregated in 500 time segments considering the past 500 segments.

samples length that overlap the previous segment by 4096 samples. At a sample rate of 44.1 kHz one segment is 185.8 ms long, with one new segment being created every 92.9 ms. The corresponding position of this segment is then searched in the other recordings in a segment of 16384 samples.

The cross-correlation is used to find the time delay between the two signals x_1 and x_2.

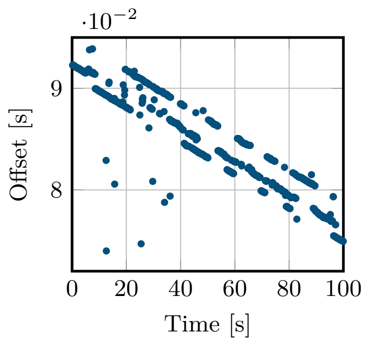
$$R_{x_1,x_2}(n) = \mathcal{F}^{-1}(X_1(\omega)X_2^*(\omega)) \tag{7.3}$$

. where X_1 and X_2 are the Discrete Fourier Transforms of the signals x_1 and x_2. The TDoA is then the delay for which the cross-correlation R_{x_1,x_2} has the largest value. The Generalized Cross Correlation (GCC) [20] introduces weights in the frequency domain of the cross-correlation to make the calculation of the cross-correlation more robust against disturbing factors like noise and reverberations.

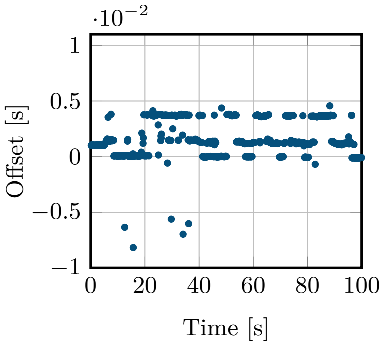
$$R^{GCC}_{x_1,x_2}(n) = \mathcal{F}^{-1}(X_1(\omega)X_2^*(\omega)\psi(\omega)) \tag{7.4}$$

One such weighting function that is used in conditions with reverberations is the Phase Transform (PHAT) [20]. It normalizes each frequency component and only uses the phase.

$$\psi_{PHAT}(\omega) = \frac{1}{|X_1(\omega)X_2^*(\omega)|} \tag{7.5}$$



(a) The raw offsets between both recordings. The slope in the graph is due to the difference in sampling rates r_k between the two participating devices.



(b) The offsets after compensating the difference in sampling rates r_k using our voting scheme. Each pair of recordings is drift compensated independently.

Figure 7.3: One dimension of D_i from a recording of a discussion with 3 participants taking turns.

This method is then called Generalized Cross Correlation with Phase Transform (GCC-PHAT). The TDoA d can the be calculated according to:

$$d = \arg\max_{n} R^{GCC}_{x_1,x_2}(n) \tag{7.6}$$

Figure 7.3a shows three different speakers taking turns in a discussion. The difference in the TDoA from time segments when one speaker is active to segments when another speaker is active are clearly visible. The slope is caused by the difference in the sampling rates of the two devices r_k.

7.2.2 Clock Drift

Experiments with different smartphones have shown that they do not record the audio at exactly 44.1 kHz. Figure 7.1 shows how quickly clock drifts aggregate to exceed the time difference of arrival values obtained rooms of regular size for meetings.

This is due to manufacturing tolerances and temperature differences. The differences measured are up to ± 15 samples per second. Without compensating these differences, two corresponding audio segments diverge and do not overlap anymore after a few minutes. Also, the TDoA vectors for any given speaker change over time if the difference in sampling rate is not compensated.

As a result of the difference in sampling rates, D_i lie on slopes as shown in Figure 7.1. The offset of these drifts is caused by different speakers being active at different times. To compute the actual propagation delay differences Δ_j that are used to detect the speakers, we need to compensate for the clock drift. Without knowledge of which speaker is active at what time, linear regression methods cannot be applied to estimate the slope. The presence of outliers makes least squares methods unsuitable. Instead, we compute the most likely slopes r using a voting scheme. For each newly D_i, we compute the slope to 500 D_j. The measurements D_j to which D_i is compared are cast from the last 2000. Each resulting slope casts a vote for the the actual slope. The binning then aggregates the most likely slope on-line by adding votes for each new TDoA measurement. The initial range that the binning spans is $-4 \ldots 4$ samples per time segment and contains 800 bins.

Figure 7.2 shows a typical binning showing a clear peak for a slope of roughly -1 samples per time segment containing 8192 samples. To more accurately estimate the slope, the range which the binning spans is reduced to accommodate the most likely slope values on-line. Previous measurements D_i are updates as the slope estimation becomes more accurate. Figure 7.3b shows the values of the slope compensated D_i $(D_i - t \cdot R)$ and the corresponding to the raw D_i values in Figure 7.3a.

7.2.3 Clustering

After accounting for the clock differences, the differences in propagation delays Δ_s are the main unknown influences on the measurements from Equation 7.2. For each time window i we can compute l_i from the initial measurement D_i:

$$l_i := \Delta_s + W = D_i - t \cdot R - C \tag{7.7}$$

The values that l_i can assume are directly related to the time differences caused by different sound sources Δ_s and the measurement noise. Hence we do expect a user to cause values of l_i that are similar regardless of the frame number i. To find the actual set of Δ_s we cluster the results of the right side of Equation 7.7. The DBSCAN [33] algorithm clearly outperformed K-means clustering due to the large number of outliers that are present in the measurements. Running DBSCAN iteratively allows us to add new measurements at run-time.

By iteratively adding new data points, the density of noise points increases. This can lead to the merging of individual clusters which may represent different speakers. To avoid this problem, the number of data points is kept constant by removing the oldest data points. Clusters vanish when they have no data points left but the position of the previous clusters is stored. When a new cluster is created, it is compared to the position of the previous clusters and is connected to it if the positions are close. Therefore, speakers that were quiet for some time can be correctly detected when they start speaking again.

The measurements D_i often contain noisy components. Since the difference in propagation delay is short for all pairs of phones assuming that the recording area is limited, we can easily filter noisy measurements. After that, most Δ_s do not contain all components. Even so, the measurements should be clustered. To achieve this, the Partial Distance Strategy [54] is used to compute the distance between two data points using all components that exist in both data points. The distance between the vectors l_i and l_j each with N dimensions is calculated according to

$$d = \sqrt{N \frac{\sum_{k=1}^{N} (l_{i,k} - l_{j,k})^2 I_k^{i,j}}{\sum_{i=1}^{N} I_k^{i,j}}} \tag{7.8}$$

with $l_{i,k}$ being the k^{th} component of l_i and

$$I_k^{i,j} = \begin{cases} 1, & \text{if } k^{\text{th}} \text{ component is defined in } l_i \text{ and in } l_j \\ 0, & \text{otherwise} \end{cases} \tag{7.9}$$

Additionally, the distance d is set to infinity when too few corresponding vector components are available after filtering. Since the geometry of the phones and the position of the speakers are not known, it is not possible to determine if these available components are sufficient to distinguish the different speakers. In the worst case clusters representing different speakers get merged. To avoid this problem, a high number *minPts* for the DBSCAN clustering is used and a penalty for measurements with only few components is introduced:

$$d' = d \frac{N}{\sum_{K=1}^{N} I_k^{i,j}} \tag{7.10}$$

7.2.4 Temporal Filtering With Markov Model

Time segments may be incorrectly classified as silence or as another speaker. These errors can occur because of short pauses or environment noise that caused the calculation of the TDoAs to give wrong results. Subsequently these time segments were associated with the wrong speaker in the clustering algorithm.

These errors can be corrected by assuming a structure for conversations that can be captured in a Markov model. For example, it is unlikely that speakers take turns 10 times per second. The states in the model we use represent the active speaker and silence. It is assumed that only one speaker is active at the same time. The transitions between the states represent the probability of moving from one state to an other in one time segment. If a speaker is active in one segment, the same speaker will probably still be active in the next time segment 92.9 ms later. Therefore, the probability of staying in the same state is higher than the probability of changing to an other active speaker or to silence. For each state there are emission probabilities describing the probability of getting a certain observation when being in this state. The observations here are the different cluster assignments. The probability of observing the cluster assignment corresponding to the current state is highest while the probability of observing silence or another cluster is assignment is smaller. The Viterbi Algorithm is used to find sequence of states $x_1, ..., x_T$ of the Hidden Markov Model that matches the observations best.

7.2.5 Distributing the Workload

The smartphones used today vary in their processing power. Therefore, it is necessary to distribute the workload of processing the audio recording pairs to the participating phones such that all can complete their work in time. Since the step size of the processed audio segments is 4096 samples, one segment should be processed in 92.9 ms. On smartphones with multiple cores, multiple segments can be processed in parallel.

After starting the application, the audio recording pairs are distributed evenly to the participating smartphones through WiFi. Each pair is processed on one of the smartphones that is part of the pair. This helps to minimize the network bandwidth utilized by *Disca*. Also, each phone monitors the time required to process one segment. If the required time exceeds the available time of 92.9 ms, it requests its neighbors to take over the computation for their respective audio pair. So the workload allocation is handled in a fully distributed manner without chaning the network bandwidth requirements. After sending a neighbor a request to pass on the responsibility of processing the pair of recordings, the other smartphone accepts or refuses depending on the available processing time. If the transfer is rejected, we try to pass on one of the other pairs of recordings in which the overburdened smartphone is involved. We observed that even dated Android devices such as the Galaxy Nexus easily handle the computational burden. After calculating the TDoA for an audio pair, the smartphone transmits the calculated value to all other smartphones. When all measurements for one time segment are received, the clustering and classifying steps are executed on each smartphone.

7.3 Evaluation

To evaluate our speaker detection system two setups have been used. Firstly, actual conversations with three speakers sitting around a desk have been recorded. In total, 4 different seating positions, rooms and combinations of people were recorded for a total of 20 minutes. The rooms were not chosen to be explicitly quiet and noise sources such as air conditioning or people moving and talking outside the open door were present. Each of these discussions was annotated manually.

To do a long term test, we used a 5.1 speaker system. We distributed an audio book such that it was played from one speaker at a time. The speaker was switched in a random pattern to simulate multiple people taking turns speaking. This setup, by design, provides an accurate ground truth about which speaker was active at which time. Like this, a total of 60 additional minutes of annotated data was obtained. All the experiments were recorded with five smartphones (Samsung Galaxy S3, Samsung Galaxy Nexus, Samsung Nexus S, HTC One M7).

7.3.1 Segment Length

The length of the audio segment has a large impact on the reliability of the observed TDoAs $d_k(i)$. With smaller segment lengths, fewer TDoAs are correctly estimated. As a result, incorrect observations are removed in the filtering step and some are classified as noise. This leads to a poor clustering result with many time segments not assigned to any speaker. However, the processing power limits the length of the segments, larger segments require more processing power to compute correlation functions. Additionally, the segments should capture only one active speaker and also detect short pauses. For the remainder of the evaluation, a segment length of 8192 was used. This allowed us to run *Disca* on our test devices in real time.

Figure 7.4 shows for the different fragment lengths, which segments could be assigned to a cluster.

7.3.2 Distributed Speaker Diarisation performance

Naively comparing the ground truth annotation to the output of our diarisation algorithm leads to roughly 93% of time segments being correctly classified.

In the real discussion experiments, performance is slightly worse at 90%. Manual inspection showed that many misclassifications occur when the

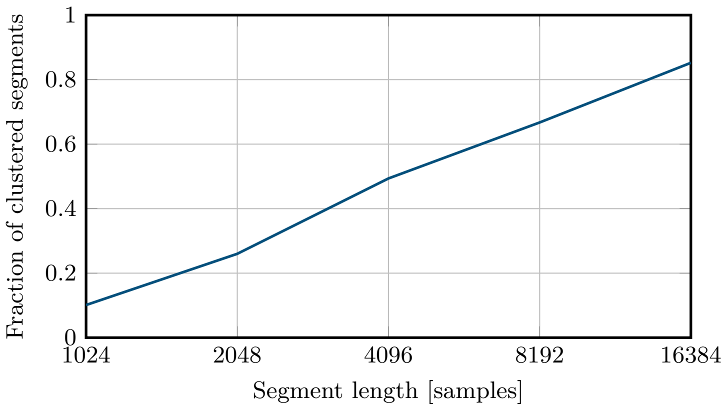


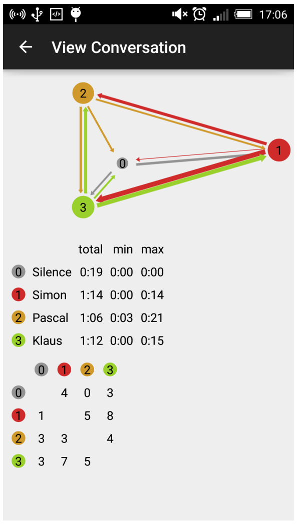
Figure 7.4: Segments that could be clustered for segment lengths from 1024 to 16384 samples. With shorter segment length fewer segments could be assigned to a cluster.

speaker changes. More explicitly, time segments that are within 0.2 seconds of a speaker change annotated in the ground truth are only in 58% of cases correct. Ignoring these time segments, 94% of the remaining time segments to be correctly classified. Note that 8% of time segments lie within 0.2 seconds of a speaker change.

When annotating the recordings, it is in many cases unclear at which time exactly a speaker changes happen. In most cases there is a slight pause between the speakers and it is unclear to which speaker the pause should be assigned. In other cases, one speaker interrupts another creating a slight overlap. In the rarest cases, speakers change without either an audible pause or overlap.

In the audio book experiment, the performance is slightly higher at 96% of the segments being correctly identified. Neglecting errors within 0.2 seconds of a speaker leads to 98% of the segments being correctly classified. Note that, again, 8% of time segments lie within 0.2 seconds of a speaker change. The improved performance is mostly due to the more controlled sequence of speakers without long pauses or overlaps. Also, the ground truth data is not subjective and free of annotation errors due to the experimental setup.

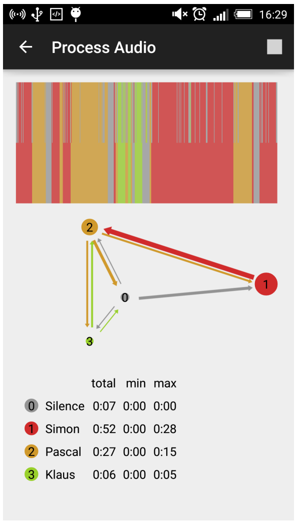
To estimate the performance when less smartphones are contributing to the system, we randomly selected three of the available five recordings. The results were within 1% of the previously discussed results with five



	total	min	max
0 Silence	0:19	0:00	0:00
1 Simon	1:14	0:00	0:14
2 Pascal	1:06	0:03	0:21
3 Klaus	1:12	0:00	0:15

	0	1	2	3
0		4	0	3
1	1		5	8
2	3	3		4
3	3	7	5	

(a) Overview of a previously recorded conversation.



	total	min	max
0 Silence	0:07	0:00	0:00
1 Simon	0:52	0:00	0:28
2 Pascal	0:27	0:00	0:15
3 Klaus	0:06	0:00	0:05

(b) On-line visualization of the speaker activity.

Figure 7.5: Selection of saved conversations and overview of those. Transitions between speakers are shown in the transition diagram and the number of transitions are displayed. For all speakers their time contributing to the conversation is listed.

recordings.

7.4 Android Application

The implemented Android application is able to detect the active speakers in real time. When starting the detection, the participating persons can be selected. Speakers selected on one phone are automatically matched to the cluster which is closest to that phone. The number of speakers is not tied to the number of phones participating. Additional speakers are assigned a color which at any time can be matched with a name manually.

While detecting the active speakers, the application shows at the top of the screen the sequence of the last speakers. The upper half of the bar in Figure 7.5b shows the result from the clustering step and the lower half the detected speakers after filtering with the Markov model. The transition diagram is updated periodically and shows how often the transition between the different speakers and silence occurred. The area of the circles corresponds to the total time the person has spoken. When the detection is stopped, the results are saved. Figure 7.5a shows a the statistics available for a completed discussion. In addition to the information shown while processing, the number of transitions is also shown in text form.

7.5 Conclusion

We have shown how a set of off-the-shelf smartphones can be used to distinguish active speakers in a conversation. We show that speaker diarization can be performed using multiple phones and software albeit the practical limitations of inaccurate clocks. As opposed to the previous chapter, this allows for unambiguous speaker annotation. The resulting system could also be used to perform beamforming to boost the audio quality for the active speaker.

8

Conclusion

In this work we have discussed how observations made with regular smartphone sensors can be used in various ways. Localization, indoors and outdoors, can be simplified and improved relying on existing infrastructure. Existing WiFi networks can be used as landmarks similarly to how people use visual objects to localize. In combination with an accurate method to track where people walk, we have shown how to create signal strength maps without hassle. Similarly, we relied on the existing GPS infrastructure to localize more quickly and robustly compared to state of the art least squares localization schemes. The receiver design we propose requires only a few milliseconds of signal to compute a position fix with minimal requirements about initial location and time guesses. As a result, signal recordings can be stored for off-line localization without processing them immediately. This is especially useful to conserve energy or to store a GPS signal snippet to geotag data such as photos. We also used smartphones to track speaker activity during discussions. The locations of the speakers translate to distinct propagation delay differences between participating devices. In order to observe the propagation delay differences in the recorded audio signals, non deterministic system delays and differences in the device clocks have to be tracked. The propagation delay differences can be observed in the recorded audio signals. We have also shown that motion data from wrist-worn devices can be used to detect gestures. Using the alphabet as the gesture set, we have shown how coordinate transformations can significantly improve recognition performance.

Albeit the benefits of accurate motion sensing are apparent, we show how such data can be exploited to infer touch input. To do so we refrained from using lab experiments and instead sourced our data from a smartphone game that has been played in unknown environments by 2'049 different players. Although the results show that touch inference is a difficult task, motion data can significantly reduce the number of guesses required to crack a password or pin. Since current mobile operating systems are not built to consider motion data to be sensitive information, touch input can be inferred by unauthorized applications. Especially because such data is entered on a regular basis.

Returning to Douglas Adams' quote - in the context of this thesis - seeing first means building a prototype, setting up a measurement system, or collecting data before there is certainty on the outcome. During my time as a PhD student, I was fortunate enough to see many ideas being pursued in the face of complete uncertainty. Some of these ideas made it into this thesis but most did not. Therein lies the risk of seeing first. Often, results are unsatisfying after a lot of work has been put into seeing if there is something to be achieved with an idea.

However, this policy of seeing first also gives insights that are not influenced by expectations or prejudice. Clearly, this is what Douglas Adams' quote is conveying and even though the cost was high, seeing first has resulted in this thesis.

Bibliography

[1] Akos, D.: A Software Radio Approach to Global Navigation Satellite System Receiver Design. PhD thesis, Ohio University, Athens, OH (1997)

[2] Anguera, X., Wooters, C., Hernando, J.: Acoustic beamforming for speaker diarization of meetings. (2007)

[3] Anguera, X., Wooters, C., Peskin, B., Aguiló, M.: Robust speaker segmentation for meetings: The ICSI-SRI spring 2005 diarization system. MLMI. Springer (2005)

[4] Ardüser, L., Bissig, P., Brandes, P., Wattenhofer, R.: Recognizing text using motion data from a smartwatch. In: Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on, IEEE (2016) 1–6

[5] Aviv, A.J., Sapp, B., Blaze, M., Smith, J.M.: Practicality of accelerometer side channels on smartphones. ACSAC (2012) 41–50

[6] Axelrad, P., Bradley, B.K., Donna, J., Mitchell, M., Mohiuddin, S.: Collective detection and direct positioning using multiple gnss satellites. Navigation **58**(4) (2011) 305–321

[7] Bahlmann, C., Haasdonk, B., Burkhardt, H.: Online handwriting recognition with support vector machines - a kernel approach. ICFHR (2002) 49–54

[8] Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): Part II. Robotics Automation Magazine **13**(3) (2006) 108–117

[9] Bearder, M., Wolfram, S.: Stephen Wolfram on Personal Analytics. MIT Technology Review (May 2013)

[10] Beauregard, S.: A helmet-mounted pedestrian dead reckoning system. IFAWC (2006)

[11] Beauregard, S.: Omnidirectional pedestrian navigation for first responders. WPNC (2007)

[12] Bieber, G., Kirste, T., Urban, B.: Ambient interaction by smart watches. In: PETRA. (2012) 39

[13] Bissig, P., Brandes, P., Passerini, J., Wattenhofer, R.: Inferring Touch from Motion in Real World Data. In: FPS. (2015) 50–65

[14] Bissig, P., Deriu, J., Förster, K.T., Wattenhofer, R.: RTDS: Real-Time Discussion Statistics. In: MUM. (2016)

[15] Bissig, P., Eichelberger, M.: Fast and Robust GPS Fix Using One Millisecond of Data. In: Information Processing in Sensor Networks (IPSN), 2017 16th ACM/IEEE International Conference on, IEEE (2017)

[16] Bissig, P., Förster, K.T., Tanner, S., Wattenhofer, R.: Distributed Discussion Diarisation. In: CCNC. (January 2017)

[17] Bissig, P., Wattenhofer, R., Welten, S.: A Pocket Guide to Indoor Mapping. WPNC (2013) 1–6

[18] Blanke, U., Schiele, B.: Sensing location in the pocket. UbiComp (2008)

[19] Brandstein, M.S., Silverman, H.F.: A practical methodology for speech source localization with microphone arrays. Computer Speech & Language **11**(2) (1997) 91 – 126

[20] Brandstein, M., Silverman, H.: A robust method for speech signal time-delay estimation in reverberant rooms. ICASSP (1997)

[21] Bunke, H., Roth, M., Schukat-Talamazzini, E.G.: Off-line cursive handwriting recognition using hidden markov models. Pattern recognition **28**(9) (1995) 1399–1413

[22] Bush, V.: As We May Think, (Reprint from 1945). Interactions **3**(2) (March 1996) 35–46

[23] Cai, L., Chen, H.: Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In: HotSec. (2011) 9–9

[24] Cai, L., Chen, H.: On the practicality of motion based keystroke inference attack. In: TRUST. Volume 7344. (2012) 273–290

[25] Cheong, J.W., Wu, J., Dempster, A.G., Rizos, C.: Efficient implementation of collective detection. In: IGNSS symposium. (2011) 15–17

[26] Chintalapudi, K., Padmanabha Iyer, A., Padmanabhan, V.N.: Indoor localization without the pain. MobiCom (2010)

[27] Choi, S.D., Lee, A.S., Lee, S.Y.: On-line handwritten character recognition with 3d accelerometer. In: Information Acquisition, 2006 IEEE International Conference on, IEEE (2006) 845–850

[28] Chong, M.K., Marsden, G., Gellersen, H.: Gesturepin: using discrete gestures for associating mobile devices. In: Mobile HCI, ACM (2010) 261–264

[29] Closas, P., Fernández-Prades, C., Fernández-Rubio, J.A.: Maximum likelihood estimation of position in gnss. Signal Processing Letters, IEEE **14**(5) (2007) 359–362

[30] Department of Defense, U.S.o.A.: Global Positioning System Standard Positioning Service Performance Specification (GPS SPS PS), 4th Edition. Technical report (2008)

[31] Durrant-Whyte, H., Bailey, T.: Simultaneous localisation and mapping (SLAM): Part I. Robotics Automation Magazine **13**(2) (2006) 99–110

[32] Dutta, D., Chowdhury, A.R., Bhattacharya, U., Parui, S.: Lightweight user-adaptive handwriting recognizer for resource constrained handheld devices. In: Proceeding of the workshop on Document Analysis and Recognition, ACM (2012) 114–119

[33] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining. (1996)

[34] Ferris, B., Fox, D., Lawrence, N.: WiFi-SLAM using Gaussian process latent variable models. IJCAI (2007)

[35] Foxlin, E.: Pedestrian tracking with shoe-mounted inertial sensors. Computer Graphics and Applications, IEEE **25**(6) (2005) 38–46

[36] Freeman, W.T., Roth, M.: Orientation histograms for hand gesture recognition. In: International workshop on automatic face and gesture recognition. Volume 12. (1995) 296–301

[37] Godha, S., Lachapelle, G.: Foot mounted inertial system for pedestrian navigation. Measurement Science and Technology **19**(7) (2008) 075202

[38] Haykin, S., Liu, K.R.: Handbook on Array Processing and Sensor Networks. (2010)

[39] Hinckley, K., Song, H.: Sensor synaesthesia: touch in motion, and motion in touch. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM (2011) 801–810

[40] Huang, J., Millman, D., Quigley, M., Stavens, D., Thrun, S., Aggarwal, A.: Efficient , generalized indoor WiFi GraphSLAM. ICRA (2011)

[41] Jarusriboonchai, P., Olsson, T., Väänänen-Vainio-Mattila, K.: User experience of proactive audio-based social devices: A wizard-of-oz study. In: MUM. (2014)

[42] Kaess, M., Ranganathan, A., Dellaert, F.: iSAM: Incremental smoothing and mapping. Transactions on Robotics **24**(6) (2008) 1365–1378

[43] Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of Fluids Engineering **82**(1) (1960) 35–45

[44] Kienzle, W., Hinckley, K.: Writing handwritten messages on a small touchscreen. In: Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services, ACM (2013) 179–182

[45] Kolly, S.M., Wattenhofer, R., Welten, S.: A personal touch: Recognizing users based on touch screen behavior. In: Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones, ACM (2012)

[46] Lee, S.W., Mase, K.: Activity and location recognition using wearable sensors. Pervasive Computing, IEEE **1**(3) (2002) 24 – 32

[47] Liang, R.H., Ouhyoung, M.: A real-time continuous gesture recognition system for sign language. In: Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on, IEEE (1998) 558–567

[48] Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uwave: Accelerometer-based personalized gesture recognition and its applications. In: PerCom, IEEE Computer Society (2009) 1–9

[49] Liu, J., Priyantha, B., Hart, T., Ramos, H.S., Loureiro, A.A., Wang, Q.: Energy Efficient GPS Sensing with Cloud Offloading. In: Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys 2012), ACM (November 2012) 85–98

[50] Liwicki, M., Bunke, H.: Handwriting recognition of whiteboard notes. In: Proc. 12th Conf. of the Int. Graphonomics Society. (2005) 118–122

[51] Lu, H., Brush, A.B., Priyantha, B., Karlson, A.K., Liu, J.: Speakersense: energy efficient unobtrusive speaker identification on mobile phones. In: PerCom. (2011)

[52] Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA (1999)

[53] Marquardt, P., Verma, A., Carter, H., Traynor, P.: (sp)iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In: ACM CCS, ACM (2011) 551–562

[54] Matyja, A., Siminski, K.: Comparison of algorithms for clustering incomplete data. Foundations of Computing and Decision Sciences (2) (2014)

[55] McCowan, I., Gatica-Perez, D., Bengio, S., Lathoud, G., Barnard, M., Zhang, D.: Automatic analysis of multimodal group actions in meetings. Pattern Analysis and Machine Intelligence **27**(3) (2005) 305–317

[56] Michalevsky, Y., Boneh, D., Nakibly, G.: Gyrophone: Recognizing speech from gyroscope signals. In: 23rd USENIX Security Symposium, San Diego, CA, USENIX Association (August 2014) 1053–1067

[57] Miluzzo, E., Varshavsky, A., Balakrishnan, S., Choudhury, R.R.: Tapprints: your finger taps have fingerprints. In: MobiSys, ACM (2012) 323–336

[58] Miro, X.A.: Robust speaker diarization for meetings. Universitat Politècnica de Catalunya (2007)

[59] Müller, M.: Dynamic time warping. Information retrieval for music and motion (2007) 69–84

[60] Nakagawa, S., Asakawa, K., Wang, L.: Speaker recognition by combining MFCC and phase information. In: InterSpeech. (2007) 2005–2008

[61] Niu, Y., Chen, H.: Gesture authentication with touch input for mobile devices. In: MobiSec. Volume 94 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering., Springer (2011) 13–24

[62] Noll, C.E.: The crustal dynamics data information system: A resource to support scientific analysis using space geodesy. Advances in Space Research **45**(12) (2010) 1421–1440

[63] Ojeda, L., Borenstein, J.: Non-gps navigation for emergency responders. In: 2006 International Joint Topical Meeting: Sharing Solutions for Emergencies and Hazardous Environments. (2006) 12–15

[64] Owusu, E., Han, J., Das, S., Perrig, A., Zhang, J.: Accessory: Password inference using accelerometers on smartphones. In: Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications. HotMobile '12, New York, NY, USA, ACM (2012) 9:1–9:6

[65] Parviainen, M., Pertila, P., Hamalainen, M.: Self-localization of wireless acoustic sensors in meeting rooms. In: Hands-free Speech Communication and Microphone Arrays (HSCMA), 2014 4th Joint Workshop on. (2014)

[66] Petho, A., Fallis, D.S., Keating, D.: ShotSpotter detection system documents 39,000 shooting incidents in the District. The Washington Post (November 2013) `http://wapo.st/16vVyx1`.

[67] Plamondon, R., Srihari, S.: Online and off-line handwriting recognition: a comprehensive survey. Pattern Analysis and Machine Intelligence **22**(1) (Jan 2000) 63–84

[68] Porzi, L., Messelodi, S., Modena, C.M., Ricci, E.: A smart watch-based gesture recognition system for assisting people with visual impairments. In: Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices, ACM (2013) 19–24

[69] Raffa, G., Lee, J., Nachman, L., Song, J.: Don't slow me down: Bringing energy efficiency to continuous gesture recognition. In: Wearable Computers (ISWC), 2010 International Symposium on, IEEE (2010) 1–8

[70] Rai, A., Chintalapudi, K.K., Padmanabhan, V.N., Sen, R.: Zee: zero-effort crowdsourcing for indoor localization. Mobicom, ACM (2012) 293–304

[71] Romano, N.C., J., Nunamaker, J.F., J.: Meeting analysis: findings from research and practice. In: System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on. (2001)

[72] Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a wii controller. In: Proceedings of the 2nd international conference on Tangible and embedded interaction, ACM (2008) 11–14

[73] Schnupp, J., Nelken, I., King, A.: Auditory Neuroscience : Making Sense of Sound. Cambridge, Mass. MIT Press (2011)

[74] Shah, M., Mears, B., Chakrabarti, C., Spanias, A.: Lifelogging: Archival and retrieval of continuous- ly recorded audio using wearable devices. In: ESPA. (2012)

[75] Skog, I., Nilsson, J.O., Handel, P.: Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems. IPIN (2010)

[76] Steinhoff, U., Schiele, B.: Dead reckoning from the pocket - An experimental study. PerCom (2010)

[77] Stinson, L.: Having a Hard Time Being a Human? This App Manages Friendships for You. WIRED (January 2015) `http://www.wired.com/2015/01/hard-time-human-app-manages-friendships/`.

[78] Sur, S., Wei, T., Zhang, X.: Autodirective audio capturing through a synchronized smartphone array. In: Proceedings of the 12th annual international conference on Mobile systems, applications, and services. (2014)

[79] Swan, M.: Sensor Mania! The Internet of Things, Wearable Computing, Objective Metrics, and the Quantified Self 2.0. Journal of Sensor and Actuator Networks **1**(3) (2012) 217–253

[80] Tsui, J.B.Y.: Fundamentals of Global Positioning System Receivers - A Software Approach. John Wiley and Sons, Inc., New York, NY (2000)

[81] Van Diggelen, F.S.T.: A-GPS: Assisted GPS, GNSS, and SBAS. Artech House (2009)

[82] Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions on **13**(2) (April 1967) 260–269

[83] Wang, H., Tsung-Te Lai, T., Choudhury, R.R.: Mole: Motion leaks through smartwatch sensors. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, ACM (2015) 155–166

[84] Wu, J., Pan, G., Zhang, D., Qi, G., Li, S.: Gesture recognition with a 3-d accelerometer. In: UIC. Volume 5585 of Lecture Notes in Computer Science., Springer (2009) 25–38

[85] Xu, C., Pathak, P.H., Mohapatra, P.: 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, ACM (2015) 9–14

[86] Xu, C., Li, S., Liu, G., Zhang, Y., Miluzzo, E., Chen, Y.F., Li, J., Firner, B.: Crowd++: unsupervised speaker count with smartphones. In: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing. (2013)

[87] Xu, Z., Bai, K., Zhu, S.: Taplogger: inferring user inputs on smartphone touchscreens using on-board motion sensors. In: WISEC, ACM (2012) 113–124

[88] Zheng, N., Bai, K., Huang, H., Wang, H.: You are how you touch: User verification on smartphones via tapping behaviors. In: Network Protocols (ICNP), 2014 IEEE 22nd International Conference on, IEEE (2014) 221–232