Title
Collaborative and Efficient Indoor Localization for Mobile Systems

Author
Martin, Paul Daniel

Publication Date
2016

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Collaborative and Efficient Indoor Localization for Mobile Systems

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Paul Daniel Martin

2016

ABSTRACT OF THE DISSERTATION

Collaborative and Efficient Indoor Localization for Mobile Systems

by

Paul Daniel Martin

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2016

Professor Mani B. Srivastava, Chair

The introduction of GPS-enabled mobile devices created in its wake a torrent of location-based applications and services. These devices are increasingly equipped with rich sensors for position and orientation, hearing, and vision, and by using their radios they can sense proximity to nearby people and devices. Considerable research over the past decades has demonstrated that these rich sensing platforms can also be used to provide reasonable location estimates in both indoor and outdoor environments.

Research into the growing field of mobile (and indoor) localization has traditionally been concerned with providing scalable, cost-effective solutions for improving position estimation accuracy. Towards this end, researchers have explored RF-based ranging techniques, sensor fingerprinting methods, vision- and optics-based approaches, and inertial dead reckoning as potential mobile positioning technologies. Each of these methods presents a unique set of challenges—e.g., nonlinear and multipath RF propagation, training and infrastructure deployments, and sensor drift. Accompanying each of these challenges are overheads in the form of energy expenditure, increased computation, or increased wireless channel congestion. Reducing these overheads is a necessary step towards enabling indoor localization techniques to permeate the mobile market.

This work focuses on methods for reducing the various overheads associated with localiza-

tion by fusing position estimates with various low power sensing modalities, such as pressure, odometry, and even time. To do so, we distill localization routines into a spatial and temporal sequence of estimates and corresponding uncertainties. These estimates may be *absolute*, as is the case with GPS or fixed anchor nodes, or *relative* as with odometry and range. Each measurement is then treated as an edge in a graph whose nodes represent position estimates across time, allowing for joint optimization of multiple device locations across multiple sensor measurements and providing a method for easily integrating additional constraints into pre-existing positioning techniques.

We divide this work into three thrusts as follows: (i) We present an architecture and formalization of collaborative indoor localization techniques as a single cohesive estimation routine. We leverage graph realization techniques as well as sensor fusion results from the sensor network literature to jointly optimize otherwise disjoint measurements, fusing absolute and relative measurements in a single optimizing routine; (ii) We explore wideband RF time-of-flight measurements as a method of providing accurate non-line-of-sight distance measurements and channel estimations between smartphones and other mobile devices, integrating these measurements to provide accurate, real-time error corrections for mobile positioning; and (iii) We construct a generic, open-source testbed upon which to analyze the algorithms and measurements employed, facilitating easier comparison and development of state-of-the-art positioning techniques and reducing the cost of obtaining measurements with corresponding ground truth observations.

The dissertation of Paul Daniel Martin is approved.

Mario Gerla

William J. Kaiser

Gregory J. Pottie

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2016

To my mother and father . . .

whose love and encouragement remain with me always

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The work described in this dissertation could not have been realized without the support of friends and family and the help of many exemplary engineers with whom I have had the great pleasure of working over the course of my stay at UCLA.

I thank Lucas Wanner for his guidance and friendship during my formative years as a graduate student as well as his help on many projects completed along the way. I also thank Bo-Jhang Ho, Andrew Symington, and Yasser Shoukry for their invaluable contributions to and countless late nights spent working on the research presented in several chapters of this dissertation.

I owe a great debt of gratitude to my advisor, Prof. Mani Srivastava, for his support and encouragement over the past years, his relentless pursuit of technological advances, and his dedication to education in general.

I thank my father for "forcing" me on so many fishing trips along the way, my sister for remaining a beacon of light in stormy seas, and my mother for her fierce love and encouragement in guiding me to where I am today.

Finally, I thank my best friend and beautiful bride, Kelley, for her unceasing love and support throughout the entirety of my graduate studies and for putting up with far too many late nights and frozen pizzas.

This work was supported in part by the NSF under awards CNS-1329755 and CNS-1143667, by the NIH Center of Excellence for Mobile Sensor Data-to-Knowledge (MD2K) under award 1-U54EB020404-01, by the U.S. ARL, U.K. Ministry of defense (MoD) under Agreement Number W911NF-06-3-0001, and by a fellowship from Qualcomm Inc. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF and Qualcomm Inc., or represent the official policies of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defense, or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

VITA

2008–2009	Teaching Assistant, University of Pennsylvania. Philadelphia, Pennsylvania.
2009	Line 6 Engineering Technician Intern. Calabasas, California.
2010	University of Southern California Autonomous Networks Research Group, Summer Intern. Los Angeles, California.
2011	B.S. (Electrical Engineering), minors in Systems Engineering and Computer Science, University of Pennsylvania. Philadelphia, Pennsylvania.
2013	M.S. (Electrical Engineering), University of California Los Angeles. Los Angeles, California.
2013	Qualcomm Corporate R&D Intern. San Diego, California.
2015	Teaching Assistant, University of California Los Angeles. Los Angeles, California.

PUBLICATIONS

Paul Martin, Andrew Symington, and Mani Srivastava. 2016. SLATS: Simultaneous Localization and Time Synchronization. Under submission to ACM Transactions on Cyber-Physical Systems (TCPS). 2016.

Bo-Jhang Ho, Paul Martin, Prashanth Swaminathan, and Mani Srivastava. 2015. From Pressure to Path: Barometer-based Vehicle Tracking. In Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments (BuildSys '15). ACM, New York, NY, USA, 65-74.

Takamasa Higuchi, Paul Martin, Supriyo Chakraborty, and Mani Srivastava. AnonyCast: Privacy-Preserving Location Distribution for Anonymous Crowd Tracking Systems. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2015). September 7-11, 2015. Osaka, Japan.

Paul Martin, Jan Medevsky, Andrew Symington, Mani Srivastava, and Stephen Hailes. Low-Overhead Gaussian- Process Training for Indoor Positioning Systems. Proceedings of the 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN). October 13-16, 2015. Banff, Alberta, Canada.

Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. 2015. PyCRA: Physical Challenge-Response Authentication For Active Sensors Under Spoofing Attacks. In-Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security(CCS '15). ACM, New York, NY, USA, 1004-1015.

Paul Martin, Yasser Shoukry, Prashanth Swaminathan, Robin Wentao Ouyang, and Mani Srivastava. Social Spring: Encounter-based Path Refinement for Indoor Tracking Systems. The 1st ACM International Conference on Embedded Systems For Energy-Efficient Buildings, BuildSys 14. November 5-6, Memphis, USA

Paul Martin, Lucas Wanner and Mani Srivastava. Runtime Optimization of System Utility with Variable Hardware. Transactions on Embedded Computing Systems (TECS) Special Issue on Parallel Programming and Run-Time Management for Embedded Systems, 2013.

Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. 2013. Non-invasive spoofing attacks for anti-lock braking systems. In Proceedings of the 15th international conference on Cryptographic Hardware and Embedded Systems (CHES'13), Guido Bertoni and Jean-Sbastien Coron (Eds.). Springer-Verlag, Berlin, Heidelberg, 55-72.

Paul Martin, Zainul Charbiwala and Mani Srivastava. DoubleDip: Leveraging Thermoelectric Harvesting for Low Power Monitoring of Sporadic Water Use. Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys). November 2012.

Zainul M Charbiwala, Paul D Martin, Mani B Srivastava, CapMux: A Scalable Analog Front End for Low Power Compressed Sensing. Third International Green Computing Conference (IGCC'12) , June 2012.

CHAPTER 1

Introduction

The introduction of GPS-enabled mobile devices created in its wake a torrent of location-based applications and services. Now, after widespread adoption, these mobile phones have become an affordable commodity—one that more than 90% of US adults possess [Res]. These mobile phones are increasingly equipped with rich sensors of position and orientation, hearing, vision, and by using their radios they can sense proximity to nearby people and to nearby devices.

Despite growing excitement about the potential impact of mobile sensing, our phones today are still largely incapable of indoor localization on a large, ubiquitous scale. The availability of indoor localization would mean more than just enabling indoor navigation capabilities in commercial complexes like malls and airports. Rather, indoor localization is poised to enable fundamentally new applications, catalyzing the development of smarter buildings and contributing to the creation of reactive indoor spaces. The simple availability of building occupancy, for instance, allows for indoor lighting and temperature control that can target and adapt to the individuals inside, both maximizing comfort and minimizing energy expenditure [ABG10]. Moreover, indoor localization is poised to profoundly change human experience by enabling location-aware augmented reality and location-based social networks, fostering a new generation of rich, interactive indoor spaces.

Research into the growing field of indoor localization has traditionally been concerned with providing scalable, cost-effective solutions for improving position estimation accuracy. In this vein, researchers have explored *infrastructure-free* solutions such as inertial dead reckoning and ambient signal fingerprinting [AY12, Ind, Fox05, TDD11, HBS14], *infrastructure retrofit* solutions, such as WiFi fingerprinting and simultaneous localization and mapping (SLAM) us-

ing pre-existing WiFi access points [FFL07, DT08], and *infrastructure-based* solutions such as visible light [KPH14, RLR14], vision [GGS04, ZN04, OTD04], acoustics [PCB00, LR12], RF ranging-based trilateration and triangulation [MHG14, ZLL14, XFM13, LDB07], capacitive touch flooring [BHW12, STS13], and many more including the vast literature on sensor network localization (see, e.g., [BT05]). Each of these methods presents a unique set of challenges, benefits, and detriments. These come in the form of RF interference and multipath effects, sensor drift, non-stationary ambient signals, etc. Overcoming these challenges often requires considerable overheads in the form of increased computational energy (e.g. GPS in the outdoor setting), cost (when dense sensor deployments or expensive sensing hardware are required), or even channel congestion when multiple wireless messages must be exchanged to adequately constrain position estimates.

Many of these overheads prove prohibitively expensive when transitioning a given positioning scheme to wide-scale adoption. On the other hand, today's smart, connected devices are equipped with a host of inexpensive and energy efficient sensors, many of which can be used to indirectly infer device position. These sensors, when combined with pre-existing and potentially more energy hungry or otherwise costly positioning schemes (such as GPS, time-of-flight range measurement, etc.) can be used to improve estimation accuracy or decrease the overheads associated with current estimation techniques. This work describes methods to enable collaboration and cooperation between multiple mobile devices and across heterogeneous sensing and estimation subsystems such that the resulting position estimate benefits from joint optimization over multiple observations, measurements, and methods. Intuitively, we distill mobile positioning mechanics into a time series of position estimates and corresponding uncertainties. These estimates may be *absolute*, as with GPS, fingerprinting, or measurements to known anchor positions, or *relative* as with odometry or range estimation. We then treat this series of estimates and constraints as a Graph Realization Problem (GRP), leveraging results from the rich literature on sensor network localization to constrain the position of unknown nodes given observation "edges." In doing so, we provide a number of services for real-time sensor fusion given both past and current observations.

1.1 Core Contributions

The core contribution of this work is a formulation of indoor localization problems as a graph realization tailored for real-time localization optimization. Though graph realization itself has been thoroughly studied in the context of static network topography estimation, its adoption to dynamic indoor environments is not straightforward, and we explore many of the challenges associated with its use in mobile sensor networks of connected smart devices. The work presented herein is divided into three components, all aimed at constructing and demonstrating a particular GRP solver to fuse two or more sensors or estimation systems. In summary, these contributes are:

- **Graph Realization Solvers for Indoor Localization**: We present a method for reimagining localization estimation as a graph realization problem. We then demonstrate a number of GRP solvers that illustrate the benefits of jointly optimizing across heterogeneous measurements, spanning implementations using least squares in C and C++ to nonlinear state estimation routines in MATLAB.

- **Demonstrations of Jointly Optimized Localization**: To demonstrate the benefit of fusing additional information with existing localization techniques, we present four bodies of work: (i) we present *Social Spring*, a method for constraining indoor navigation systems using pairwise distance measurements between mobile users; (ii) we present a method for using relative motion constraints to aid in fingerprint construction and training using Gaussian Process ($\mathcal{GP}$) maps; (iii) we demonstrate how pressure collected from mobile barometer sensors can aid in path reconstruction; and (iv) we demonstrate how time and location may be optimized jointly in SLATS—simultaneous localization and time synchronization—an architecture for improving ultra-wideband positioning schemes by integrating models for clock errors and time asynchrony.

- **An Open-Source Localization Testbed**: Accurately evaluating a given indoor localization technology can be difficult, due in large part to the difficulty involved with collecting

ground truth positions from multiple mobile devices, deploying new sensors, measuring sensor power to perform lifetime analysis, and even providing power and communication to multiple, networked sensors. To combat these difficulties, we present an open-source testbed for providing accurate position estimates for mobile devices as well as power measurements and debugging interfaces for developing new localization infrastructures.

1.2 Related Work in Graph Realization

The field of graph realization and the application of GRP solvers to sensor networks is a well-researched one. Traditionally, GRP has seen use as a tool for reconstructing the topology of a wireless sensor network given (potentially sparse or noisy) pairwise distance measurements or angular measurements between a subset of nodes in the network or between nodes and a few anchor points. These methods can roughly be divided into (i) multidimensional scaling methods (MDS), (ii) simultaneous localization and mapping (SLAM), (iii) semidefinite programming (SDP), and (iv) graph decomposition methods. Research into MDS-based graph realization includes distributed topology reconstruction using connectivity matrices [SR04], eigenspace tracking of multiple mobile nodes [MA07], and trajectory estimation based on dynamically weighted MDS with sparse connectivity graphs [CTS07a]. SLAM methods and other cooperative range-based methods include leveraging ultra-wideband for accurate ranging in multipath environments [WLW09], offline methods for graphical methods of reducing graph dimensionality [TM06], and SLAM based on relative pose graphs for multi-robot systems [KKF10]. SDP methods include reducing positioning error for noisy measurements and sparse anchor deployments [BY04], and graph decomposition methods include spectral graph decomposition (SGD) for node-node distance measurements or distance measurements to spatial events [BLP06] as well as eigen vector approaches for optimal visual / drawing layouts with applications to graphical optimizations [Kor03]. In addition, Patwari *et al.* give an overview of time-of-arrival, angle-of-arrival, received signal strength, and ultra-wideband methods and theoretical bounds on their estimation accuracy via derivation of a Cramér-Rao bound [PAK05]. Symington *et. al*

demonstrates how various methods described above may be used to constrain estimated pedestrian trajectories in real-time localization systems (RTLS) based on mobile-to-mobile distance measurements [ST12], and [CBA10] presents an 'electronic escort service' based on pedestrian paths estimated from spatio-temporal encounters between people.

The work presented herein describes a graph realization approach for constraining location and trajectory estimates provided by potentially multiple localization routines and additional measurements. This approach is built on top of rather than in lieu of the approaches described above with extensions for real-time localization. It extends these approaches to allow for efficient solution-finding in the face of multiple mobile nodes, distance estimation errors, and variable uncertainties. More specifically, the work presented demonstrates how several of the above approaches can be extended to not only improve location estimation *accuracy* but also to reduce the overhead of the estimation routine itself.

CHAPTER 2

An Indoor Localization Primer

Indoor localization (or indoor positioning, indoor navigation, or simply mobile positioning) describes the process of estimating the position(s) $\boldsymbol{p}_i(t) = [x_i,\ \ y_i,\ \ z_i]^T$ of one or more unknown mobile nodes n_i (devices, people, etc.) based on a time-sequence of measurements. These estimates can be constrained to $\mathbb{R}^2$ or can exist in $\mathbb{R}^3$ for multi-level indoor environments or for precise positioning requirements within single-story complexes. Measurements often used for localization include RF time-of-flight (TOF), angle-of-arrival (AOA), time-difference-of-arrival (TDOA), received signal strength (RSS/RSSI), inertial/pedestrian dead reckoning (PDR), signal fingerprinting (e.g., magnetic, acoustic, RF), and many more. Furthermore, these solutions may be infrastructure-free, may require additional infrastructure, or may leverage pre-existing infrastructure such as WiFi access points.

Providing an accurate, low cost estimate of sub-room indoor positioning remains an active area of research with applications including reactive indoor spaces, dynamic temperature control, wireless health, and augmented reality, to name a few. Recently proposed indoor localization solutions have required anywhere from zero additional infrastructure to customized RF hardware and have provided room-level to centimeter-level accuracy, typically in that respective order. One example technology that has proven a pragmatic solution for scalable indoor localization is that of Bluetooth Low Energy beaconing, spearheaded by Apple's recently introduced iBeacon protocol. In what follows, we use localization tools developed around the iBeacon protocol as an introduction into indoor and mobile positioning methods, providing an in-depth look at Bluetooth Low Energy's viability as an indoor positioning technology and motivating the need for more accurate estimates in some scenarios. This particular system demonstrates an

average position estimation error as low as 0.53 meters with sufficient deployment density, and it serves to illustrate one of two major localization techniques—those based on wireless signal strength—with a followup discussing another major category, those based on precise timing of signal propagation. Both of these techniques have important drawbacks, motivating the need for tertiary information in order to either improve their accuracy (as is the case with signal-strength methods) or reduce their power consumption (for time-of-flight ranging).

2.1 Localization by Signal Strength

Wireless signal strength may be used in order to estimate the range of a receiver with respect to a transmitter, given that the transmit power itself is known. This is often done using a received signal strength indicator, or RSSI. One popular technology for RSSI-based localization is Bluetooth low-energy (BLE or BTLE). BLE peripheral devices announce themselves to nearby devices through advertisement packets, typically sent at fixed intervals for a short duration. These packets are broadcasted and thus do not require a paired connection. Because of this, advertisement packets can be used to send low energy, low throughput data from transmit-only "beacons." These beacons typically contain unique identifying information as well as calibrated signal strengths at a 1 meter distance, allowing receiving devices to estimate distances to transmitters and to obtain beacon positions via externally maintained databases.

As an example, we have created a system that treats mobile devices as naïve sensors, relaying overheard Bluetooth LE beacons to a server for subsequent processing. The server uses additional knowledge about each beacon's spatial mapping to generate an estimate of the mobile user's location. More specifically, our system follows the flow depicted in Figure 2.1 and described below.

A mobile phone listens to Bluetooth LE advertisements in a background service, estimating and relaying aggregated beacon distance estimates and corresponding beacon identifying information to a central server at fixed intervals. These distance values are estimated from received signal strengths denoted by P_{RX} and calibrated signal strengths at a 1 meter distance, denoted

for simplicity by P_{TX}, by the equation

$$\hat{d} = exp[a \cdot (P_{RX} - P_{TX})] \tag{2.1}$$

where $\hat{d}$ is the estimated distance between the phone and the transmitter and a is a pre-calibrated exponential decay term. In the case of Apple iOS devices, this conversion is done internally for iBeacon devices. For other devices like the Android tablets used in this work, it must be calculated explicitly for each received packet. The server receives these estimated beacon distances and assigns a certain weight $w(\hat{d})$ to each such that larger distance estimates (which are inherently more noise-prone) have less bearing on the final position estimate; in practice, we have found that using an exponentially decaying weighting function $w(\hat{d}) = 1/\hat{d}^{\epsilon}$ with ϵ between 1 and 1.5 performs the best. The optimal weighting exponent depends largely on, among other things, P_{TX}. The server searches for an instantaneous position estimate $\hat{x} \in \mathbb{R}^2$ by solving the equation below

$$\hat{x} = \arg\min_{x} \sum_{b \in B} w(\hat{d}_b) \| d_b - \hat{d}_b \|^2 \tag{2.2}$$

where $d_b = \|x - x_b\|$ and B is the set of overheard beacons. Using Equation 2.2 in an experiment in a 9×10 m^2 laboratory with proper post-processing filtering, we obtain an average error of 0.53 meters. This system was demonstrated at the Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings [MHG14] in a 30 m^2 area, where participants walked around with multiple mobile devices located in real time by four Bluetooth beacons.

Though signal strength based localization methods like the one described above using BTLE beaconing can be quite practical and accurate given a high enough deployment density, they suffer greatly from nonlinear RF propagation dynamics, multi-path, and fading effects. Because of this, it can be quite challenging to map a given received signal strength to an accurate distance estimate, and it can be even more challenging to map a set of received signal strengths (received from multiple beacons) to a position in 2D space. Figure 2.4 demonstrates that, among other

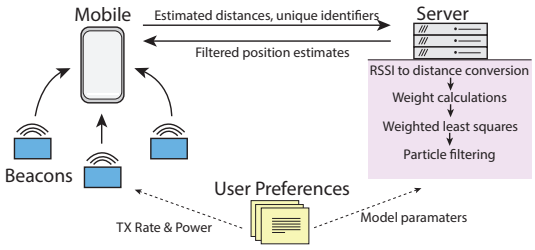


Figure 2.1: Bluetooth LE position estimation flow

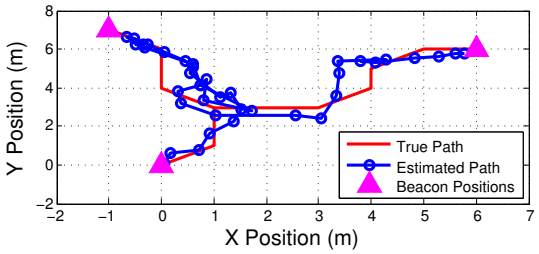


Figure 2.2: Example estimated and true path with three anchors

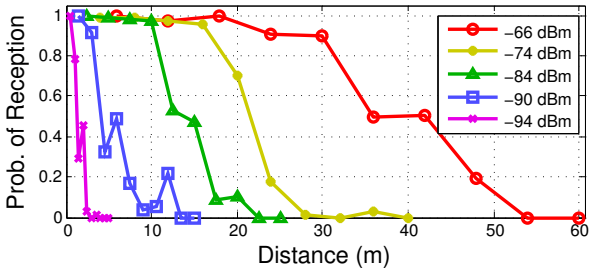


Figure 2.3: Beacon reception probability vs. distance

things, this estimation accuracy depends greatly on the transmit power of the beacons themselves. To overcome these downsides, many more accurate ranging and localization methods for indoor spaces leverage RF packet timing, like below. This, however, comes at the cost of dramatically increased power consumption. While BLE beaconing devices may last for years on small coin cell batteries, those performing time-of-flight ranging with comparable frequencies might last for days or less.

2.2 Localization by Time of Flight

One method of obtaining more accurate distance measurements is through measuring signal time-of-flight (TOF), also called signal propagation time, propagation delay, or (equivalently) one-half round trip time (RTT). To perform time-of-flight measurement, two nodes must communicate with each other in a bi-directional manner[1]. One node, A, begins by sending a message m_0 to node B at time t_0. node B receives m_0 at t_1 and replies at time t_2. The reply message, m_1,

[1] Alternatively, communication can be unidirectional if the nodes are tightly time synchronized—this is called time-of-arrival and will be discussed in detail in later chapters.

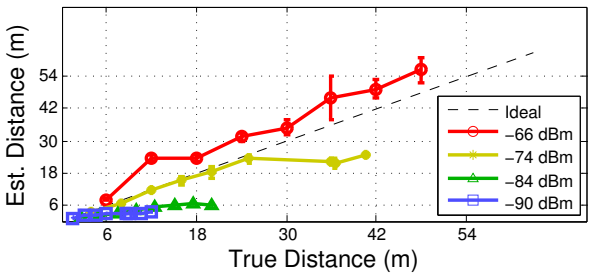


Figure 2.4: Distance accuracy vs. TX power

is received at node A at time t_3. Round trip time can be estimated by node A as $t_3 - t_0$, including 2x propagation delay as well as processing delay at node B. If this processing delay is deterministic and calibrated as, say, t_B, then propagation delay can be calculated as $T_p = \frac{t_3 - t_0 - t_B}{2}$. Given the propagation velocity v of the signal in question (e.g. $v = c$, the speed of light, for RF signals), the distance can be estimated as $\hat{d} = v \cdot T_p$. Armed with pairwise distance estimates, we can again use a number of optimization routines, graph realization methods, topology estimation, etc. to obtain a position estimate for an unknown node, say B. In its most basic form, we can again use trilateration based on distance measurements from 3 or more known anchor nodes to estimate the position of B. Though distance measurements based on time-of-flight are considerably more reliable than those based on signal strength, they still suffer from multi-path and fading effects and therefore benefit from modeling as we will see in future sections. Additionally, performing accurate RF time stamping typically requires more expensive and more power-hungry hardware, limiting its use in commodity and battery powered devices.

2.3 A Case for Joint Optimization in RTLS

Signal strength based methods are appropriate when cheap, quick deployments are required and high accuracy is not of utmost concern. Time-of-flight can offer a greatly improved position estimates, but in many cases these methods prove infeasible due to additional infrastructure requirements and increased power overheads. Considerable work like that demonstrated in [ST12] demonstrates methods for fusing one or more measurements (in [ST12], the measurements are inertial dead reckoning and pairwise distance measurements). Additionally, a

large subset of dynamic bayesian networks (DBN) such as particle filters and Kalman filters are designed explicitly to accommodate heterogeneous measurements with variable certainty and noise models. Though these filters and models are often used with great success in fusing measurements to improve the accuracy of a given positioning scheme, the goal of sensor fusion in this regard has traditionally been that of increasing position estimation accuracy. This work attempts to demonstrate ways in which many of these fusion techniques—e.g. graph realization and Kalman filters in particular—can be used not only to increase accuracy but also to decrease power consumption, decrease RF congestion, and decrease deployment overheads. In the following chapter, we will describe a structured way of using graph realization to augment a given positioning scheme with additional measurements and constraints, and we will bolster this idea with concrete implementations in the remaining chapters.

CHAPTER 3

Localization as a Graph Realization Problem

Classically, graph realization problems (or GRP for short) take as input (noisy, possibly corrupted) pairwise distances between nodes, δ_{ij}, and the goal of GRP is to recover the positions of each node $\boldsymbol{p}_i \in \mathbb{R}^k$ such that the distances ($||\boldsymbol{p}_i - \boldsymbol{p}_j|| \cong \delta_{ij}$) are satisfied, and if so the graph is said to be realizable in $\mathbb{R}^k$. This problem is known to be NP-hard, in general. Additional assumptions such as restricting to Euclidean distance measures, limiting recovery to $\mathbb{R}^2$, and allowing for some small tolerable error ϵ improves the situation, allowing for polynomial time algorithms [Hen92]. Moreover, tractable solutions to the GRP are known when given bearing, θ_{ij}, between nodes instead of distances. Contemporary methods have proven quite robust in the face of noisy inputs, permitting tolerable recovery even in cases where 90% of edges consist of corrupted measurements [Sin11, CLS12]. The work in this dissertation is motivated by the fact that GRP methods are principled, benefit from extensive theoretical analysis, and promise to enable robust recovery of indoor position estimates.

In sensor networks, RF or acoustic ranging is typically employed to provide pairwise distances; similarly RF-based angle-of-arrival or acoustic time-difference-of-arrival can provide bearings. In the context of indoor mapping, though, we have much more information than what is assumed by GRP formulations for sensor network localization (SNL). For example, by treating location fingerprints or unknown mobile positions as nodes, we can use dead-reckoning to estimate both pairwise distances and bearings between pairs of nodes. Using many such traces via crowdsourced or participatory sensing, one direct method is to average the distances and bearings ascertained from each trace to populate edges of a single aggregate graph. This can then be formulated as a GRP to recover positions of each node, i.e. to localize each fingerprint.

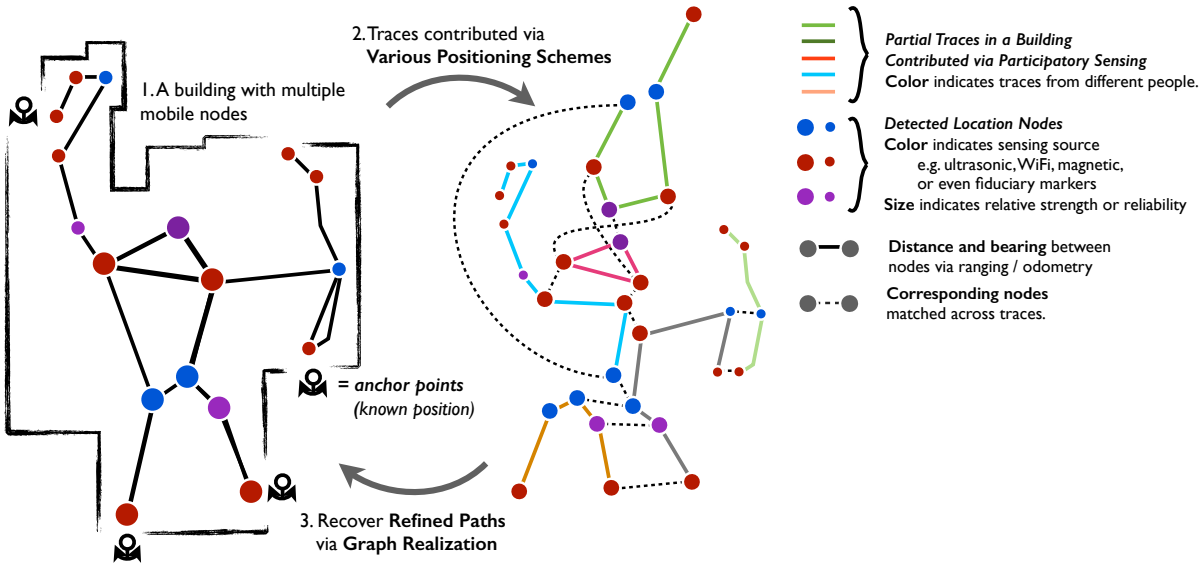


Figure 3.1: Graph realization of multiple path estimates from multiple users.

Simply aggregating traces in this manner is a rather naive approach, but useful to establish that indoor mapping can be realized as a GRP.

There are sources of noise and error particular to indoor localization that invalidate the assumptions of many existing GRP methods. GRP usually assumes that the set of nodes is given and fixed, and this assumption works in SNL since in that context sensor nodes have a unique and well-defined identity. For indoor positioning, each "node" may represent only a probable location—it might be spurious and it may not uniquely correspond to a location. Moreover, bearing and distance estimates between location estimates (nodes) are often not completely corrupted, but rather incur predictable errors due to dead-reckoning etc. Together, these uncertainties can be modeled as a GRP with weighted nodes and edges. It seems then that GRP gives a principled way of reasoning about the problem of optimizing indoor localization systems (and indoor mapping, for that matter). In this work we formulate GRP methods specifically tailored for the problem of mobile positioning, accounting for the nuances and nuisances present in the collected data and measurements from multiple systems and devices.

Inspiration for this work is rooted in part in the rich literature on simultaneous localization and mapping (SLAM), a concept developed to localize robots moving through unknown spaces. The typical goal is to establish a map as quickly as possible using high-resolution, expensive

sensors like cameras, LIDARs, and high-grade inertial sensors (IMUs) so that localization can be performed immediately from the start. Cooperative SLAM or decentralized SLAM, and SLAM with inexpensive IMUs on humanoid robots are active areas of research in the robotics community that seek to incorporate data available from many robots or data from humanoid robots, respectively. WiFiSLAM [FFL07] and FootSLAM [AR12] are two such methods proposed specifically for indoor localization. A key requirement of any successful positioning technology is that it scale to many participants (with many more contributed traces), requiring fundamentally scalable techniques for mapping and positioning from the beginning. We believe that a GRP formulation, given its deployment in large-scale SNL settings, provides that much-needed scalability.

Finally, with multiple users contributing potentially sensitive location information it is important to consider user privacy. Although the data collected form users and their mobile devices can itself be anonymized, certain sensing modalities used for location fingerprinting carry the risk of inadvertently sharing other sensitive information. For instance, audio spectrum data has been proven as a useful marker of location, but collecting *too much* of the audio spectrum at a high sampling rate or at low granularity can leak things like speech episodes or even the contents of private conversations. Additionally, long-term sharing of movement patterns opens up the possibility of behavioral modeling. Although ideally we would never deliberately try to make such sensitive inferences, we still believe that privacy-aware system design can significantly ameliorate privacy concerns. For example, we can create systems that upload data in batch so that a user can randomize the ordering of events thereby reducing our ability to perform behavioral inferencing. These and other precautions lead to information-efficient system designs that ensure some level of user privacy and increase transparency about what exactly is collected, ensuring that users do not abandon our sensing campaigns due to privacy concerns [Shi09, RES10]. While the work described in this dissertation does not directly deal with these issues of privacy, they remain nonetheless important when considering wide-scale localization technologies like those described in subsequent chapters.

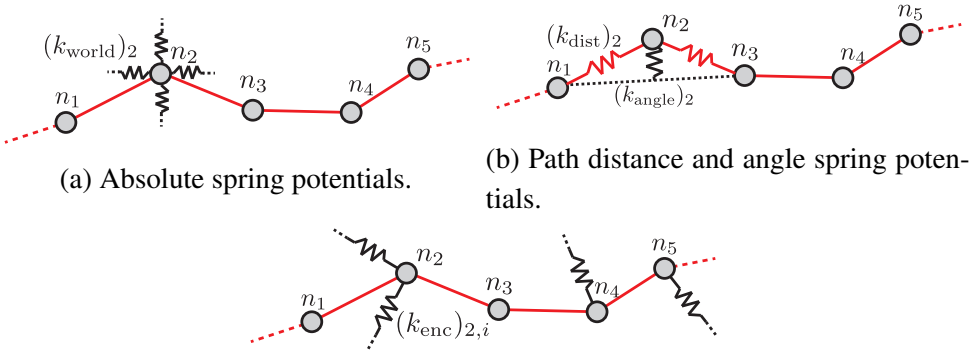
What follows are several bodies of work that demonstrate GRP-based methods for and re-

sults from joint optimization over multiple measurement types on mobile devices. Each work demonstrates that the fusion of low-overhead, low-power sensing modalities into more traditional (and potentially higher power) positioning technologies serves to reduce power, reduce RF overheads, or decrease the burden of deploying and calibrating existing positioning techniques.

CHAPTER 4

Graph Realization from Spring Potentials

Graph realization allows for network topology reconstruction based on a number of edgewise constraints. One natural source of these constraints is that of relative distances between multiple users—that is, *encounters* between two or more people at given points in time. Recent work has explored these more *social* aspects of localization. Oftentimes these encounters take the form of relative distance estimates between nearby people via acoustics, peer-to-peer radios, or other emerging mobile technologies. Of particular note are Social-Loc [JGC13] and Encounter Based Tracking [ST12]. Social-Loc operates by strategically pruning particle filter estimates, reducing the errors of current and future positional estimates in real time. Encounter Based Tracking treats encounters as constraints in a graph realization problem, minimizing positional estimates in an offline manner. This chapter describes Social Spring, a generalizable system that sits on top of an underlying indoor positioning system in an attempt to refine position estimates by incorporating pairwise distance measurements between multiple users. The main idea behind



(a) Absolute spring potentials.

(b) Path distance and angle spring potentials.

(c) Encounter spring potentials.

Figure 4.1: Absolute, relative (path distance and angle), and encounter spring constraints acting on nodes along a path (red).

Social Spring is that pairwise distance measurements are treated as a network of springs, and positional estimates are treated as nodes with local potential functions that drive them to a lower energy (and lower error) state. The result of this formulation and the core contribution of Social Spring is a distributed position refinement algorithm that can incorporate both physical constraints (e.g. walls) and relative constraints (e.g. path shape) in an accurate and efficient manner. This can be thought of as an extension to traditional GRP solvers that allows path reconstruction to consider both inertial estimates as well as building structures. We provide both simulations and preliminary tests on real indoor localization datasets in order to lend credence to Social Spring's effectiveness over a range of environmental factors, demonstrating between 10% and 30% error reduction compared to *de-facto* inertial path estimation strategies.

4.1 Social Spring

Social Spring uses relative distance measurements between people (such as those provided by acoustics, RSSI ranging, or time-of-flight) to reduce the estimation errors of an underlying localization scheme. The underlying scheme is assumed to provide initial (potentially erroneous) path estimates as a set of vertices with coordinates $\{x_1^0, \ldots, x_{\max}^0\}$. Together, these vertices and the edges that connect them make up a graph, denoted by $G = (V, D)$, where $V = \{n_1, \ldots n_{\max}\}$ is the set of vertices, or nodes, and D is a symmetric matrix of pairwise distances between nodes in V such that element (i, j) of D, denoted by $d_{ij} = d_{ji}$, represents the distance between nodes n_i and n_j. For each node n_i we associate the position vector $x_i \in \mathbb{R}^2$. Given the information embedded in D, the goal is to assign a position for each node $x_{i \in |V|}$ so as to minimize the following [ST12]:

$$\arg\min_{x} \sum_{(i,d)\in D} (\hat{d}_{ij} - d_{ij}^*)^2 = \sum_{(i,d)\in D} (||x_i - x_j|| - d_{ij}^*)^2 \tag{4.1}$$

where $\hat{d}_{ij}$ is the distance between variables x_i and x_j and d_{ij}^* is the *measured* distance between n_i and n_j. A number of traditional methods such as least squares and multidimensional scal-

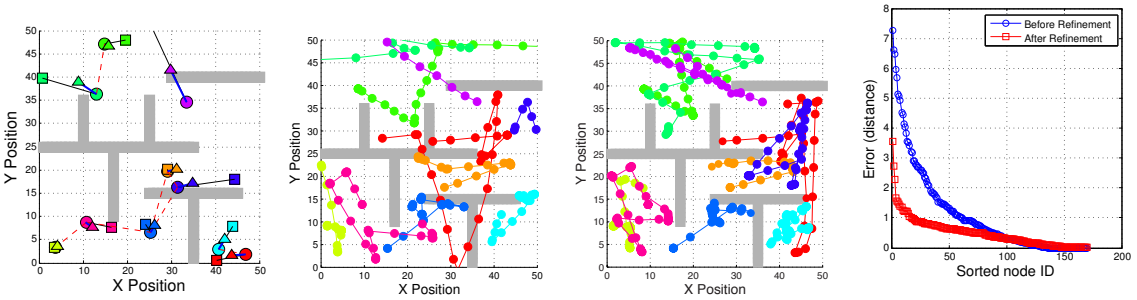
ing can be used to solve Problem 4.1 [ST12], but these generic solvers are sub-optimal in the context of indoor positioning; more specifically, these traditional methods lack three important components for indoor positioning solutions:

- An ability to cope with walls and other physical constraints imposed by the environment
- An ability to preserve some of the initial path shapes and initial location estimates
- A distributed, scalable solution

Inspired by these requirements, we turn to a simple mechanical structure that can be reasoned about in a distributed sense and whose dynamics allow for path and position preservation with varying weights—the spring. Hooke's Law states that the reactive force caused by a spring is linearly dependent on the spring's displacement from a set point. This is reminiscent of the stress term $s_{ij} \triangleq \hat{d}_{ij} - d^*_{ij}$ in Problem 4.1. In fact, instead of solving (4.1) directly, we can replace each edge $(i, j) \in D$ with a *virtual* spring of length d^*_{ij} and spring constant k_{ij} such that positional estimates are *pulled* or *pushed* towards a lower energy, lower error state. Estimated paths are augmented with absolute positioning springs (Figure 4.1a), inter-node distance springs, and node angle springs (Figure 4.1b), with the former preserving the initial location estimate and the latter two preserving the initial path shape with some (variable) certainty. In the absence of encounters, the net force from the springs depicted in Figures 4.1a and 4.1b is zero. With encounters between users, additional springs constrain the distances between nodes across paths (Figure 4.1c), resulting in a non-zero net force on the nodes.

4.1.1 Local Vertex Potentials

If K users traverse a shared space independently, each generates an isolated graph $G_{k\in\{1,\ldots,K\}} = (V_k, D_k)$ consisting of a single simple path denoted by $p_k = \{n^k_1, \ldots, n^k_{\max}\}$ with coordinates $\{x^k_1, \ldots, x^k_{\max}\}$ representing that user's estimated trajectory. Here the superscript k specifies that a node belongs to path p_k. For each node n^k_i (abbreviated below as n_i for notational clarity), Social Spring calculates the forces depicted in Figure 4.1 as follows. The initial estimates of some nodes from the underlying localization scheme may have higher confidence than others in



(a) Snapshot of Social Spring, showing true, estimated, and final positions.

(b) Estimated paths for 10 agents (before Social Spring optimization).

(c) Refined paths for 10 agents (after Social Spring optimization).

(d) Errors before and after Social Spring, sorted by descending magnitude.

Figure 4.2: Simulated user trajectories, refined estimates, and corresponding error reduction using Social Spring.

an absolute sense. In order to preserve the initial position estimate for node n_i (denoted by x_i^0), the absolute (or World) spring forces are given by

$$F_i^W = -(k_{\text{world}})_i(x_i - x_i^0) \tag{4.2}$$

where k_{world} is the spring constant whose value is proportional to the confidence with which the initial estimate x_i^0 is made. Similarly, the initial "shape" of the estimated path in terms of relative angles and distances should be preserved with some confidence. If the initial estimate for distance along path p_k from node n_i^k to n_j^k is d_{ij}^0, then the Relative distance spring force from n_i to n_j is given as

$$F_{ij}^R = (k_{\text{dist}})_{ij}(\hat{d}_{ij} - d_{ij}^0)v_{ij} \tag{4.3}$$

where $v_{ij} = \frac{x_j - x_i}{||x_j - x_i||}$ is a normalized vector from x_i to x_j and k_{dist} is the relative distance spring constant. Finally, if the initial angle incident on n_i is denoted by θ_i^0, the Angular spring force is given by

$$F_i^A = -(k_{\text{angle}})(\hat{\theta}_i - \theta_i^0)v_i^\theta \tag{4.4}$$

where v_i^θ is a vector originating at n_i and orthogonal to $x_{i+1} - x_{i-1}$. Without encounter constraints between users, $F_i^W = F_{ij}^R = F_i^A = 0 \;\; \forall \; i \in V_k, \; (i,j) \in D_k$, and all nodes are in

equilibrium.

If users can *encounter* each other, we have additional information in the form of inter-path pairwise distance measurements D_{enc}, where the element (i, j) of D_{enc} is denoted by $d_{ij} = d_{ji}$ for $n_i \in p_k,\ n_j \in p_l,\ k \neq l$. Each graph G_k no longer exists in isolation but rather is part of an aggregate graph $G = (V, D)$, where $V = \cup_{k=1}^{K} V_k$ and $D = \left(\cup_{k=1}^{K} D_k\right) \cup (D_{\text{enc}})$. The new encounter distance measurements contained in D_{enc} are used to form Encounter spring forces, denoted by

$$F_{ij}^{E} = (k_{\text{enc}})_{ij}(\hat{d}_{ij} - d_{ij}^{0})v_{ij} \tag{4.5}$$

for $(i, j) \in D_{\text{enc}}$, where d_{ij}^{0} and v_{ij} are defined as in Equation 4.3.

At each node n_i, the aggregate of these four forces, $F_i = F_i^W + F_{ij}^R + F_i^A + F_{ij}^E$, acts as a local vertex potential driving the coordinate x_i into a lower error state. Social Spring operates by iterating in small time steps until the forces acting on each node have reached an equilibrium, after which the coordinates $x_{i \in V}$ represent the refined position estimates. This allows the minimization routine to incorporate physical constraints, such as walls, at each time step, resulting in a more feasible solution. Inspiration for this method is drawn in part from distributed solutions to the "node coverage" problem [HMS02], where local potential functions are used to optimize sensor network topology.

4.1.2 Results

In order to evaluate the effects of spring constants, measurement errors, and encounters on the performance of the Social Spring refinement architecture, we developed an inertial dead reckoning simulator. This simulator mimics a number of independent users walking around a confined space by selecting random waypoints at random intervals. Each simulated user has a corresponding positional estimate arrived at by integrating a simulated, noisy acceleration signal $x = \iint_t \ddot{x}(t) + n(t)dt,\ \ x \in \mathbb{R}^2,\ \ n(t) \sim \mathcal{N}(0, \sigma_n)$. Additionally, nodes within a distance less than an encounter radius r_e at a given time will encounter each other with a certain probability $Pr(enc)$. Finally, walls can be added to simulate a more realistic indoor space, where certain

paths are infeasible.

A snapshot of the true, estimated, and refined positions of an example 10-user simulation is shown in Figure 4.2a by the circles, squares, and triangles, respectively, along with encounters represented by dashed red lines. The corresponding estimated and refined paths are shown in full in Figures 4.2b and 4.2c with the resulting error reduction shown in Figure 4.2d—an average of 56% for this example. Note in particular how paths in Figure 4.2c do not cross walls while those in Figure 4.2b do. This results in more feasible, lower error final estimates.

The percentage error reduction offered by the proposed strategy is heavily dependent on such factors as spring constants, initial estimate errors, encounter radius, and encounter probability. We explore Social Spring's sensitivity to a number of these parameters below by averaging the estimation error reduction over a number of random simulations and a range of parameter values. It is important to note that distance estimates are assumed ideal in these simulations, decoupling the errors of Social Spring from those of the underlying distance measurement mechanisms.

Study: Estimation Noise

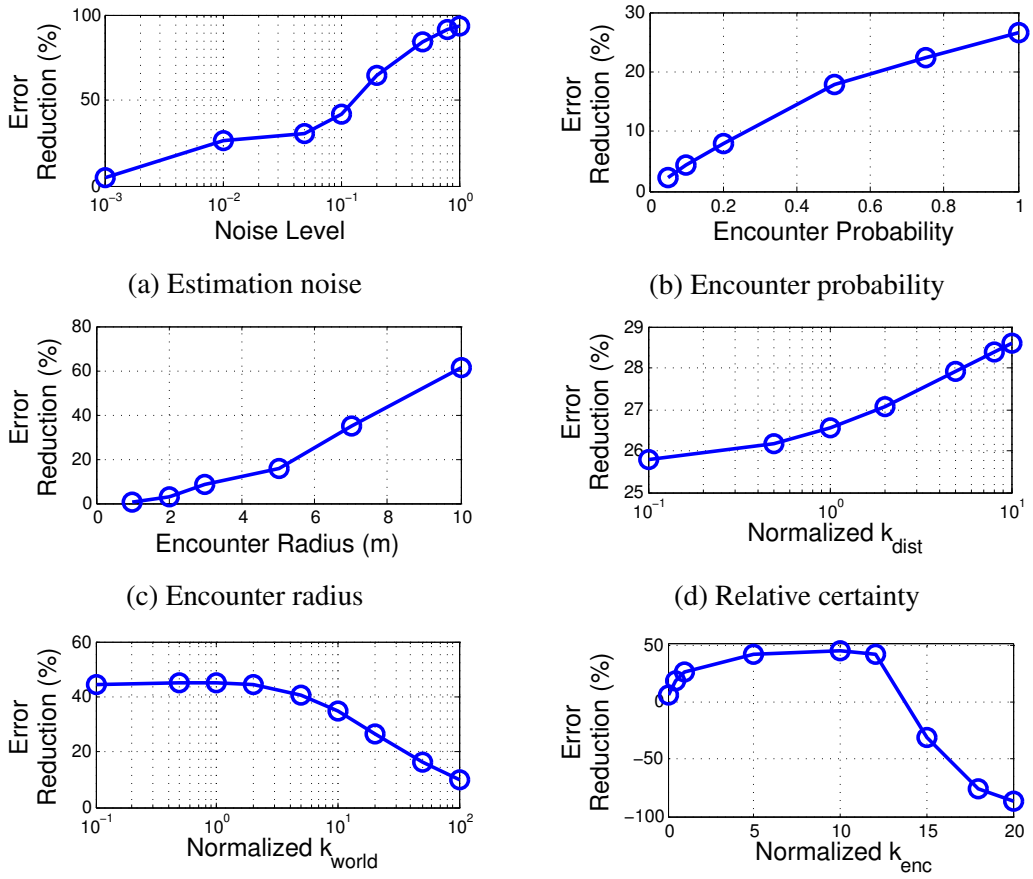
As estimation noise increases, the information provided by encounters between individuals becomes increasingly valuable. Figure 4.3a reinforces this fact—when the error is low (and the initial estimate is good) there is little to be gained from incorporating encounters, but as the error increases the percentage error reduction increases monotonically as well.

Study: Encounter Probability & Radius

Until now we have assumed that if two users are within a distance r_e of each other, they will "encounter" each other. In reality, this encounter may occur with probability $Pr(enc) \in [0, 1]$ and without perfect accuracy. As shown in Figures 4.3b and 4.3c, the performance of Social Spring depends almost linearly on both the encounter probability and the encounter radius.

Study: Relative and Absolute Certainties

The proposed path refinement scheme attempts to keep distances between nodes in a path close to their original estimates, assuming that these initial distances can be measured with moderate accuracy if the nodes are closely spaced. Additionally, certain nodes have higher absolute cer-



(a) Estimation noise

(b) Encounter probability

(c) Encounter radius

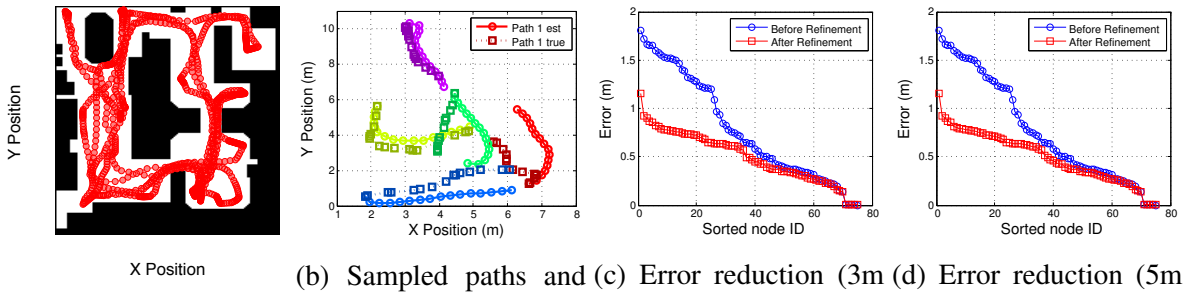
(d) Relative certainty

(e) Absolute certainty

(f) Encounter certainty

Figure 4.3: Initial estimation error reduction as a function of various system parameters and variables

tainties than others (e.g. those closer to doors where GPS is available). The results of a number of random simulations indicate that relative distance constraints k_{dist} have little effect on the overall improvement as long as they are adequately large, as shown in Figure 4.3d. Contrarily, Figure 4.3e shows that large values of k_{world} begin to degrade performance. Intuitively this is due to a "stubbornness" in the initial estimate, where nodes remain unwilling to move even with additional information provided by encounters. Finally, Figure 4.3f shows that giving additional weight to encounter information by increasing k_{enc} improves accuracy to a certain extent, but excessively high values begin to greatly degrade performance due to a breakdown in the initial graph topology and an overdependance on encounter information.



(a) Ground truth paths (b) Sampled paths and estimates (c) Error reduction (3m encounters) (d) Error reduction (5m encounters)

Figure 4.4: Results from preliminary deployment and simulated encounters.

4.1.3 Preliminary Deployment

We implemented a dead reckoning algorithm on a Nexus 7 tablet as well as the spring-based path refinement algorithm on a server with Matlab in order to evaluate Social Spring on a real dataset. A single user walked around a laboratory environment for 5 minutes, after which the 5 minutes were divided into 5 random, non-overlapping segments to simulate a multi-user scenario. The ground truth locations of the user (shown in Figure 4.4a) were manually recorded using video surveillance. Figure 4.4b shows an example of 5 randomly selected paths and their corresponding estimates via dead reckoning. Encounters were simulated offline for both $r_e = 3$ m and $r_e = 5$ m. The resulting error reductions (shown in Figures 4.4c and 4.4d) for 5 randomly selected paths averaged 15.7% and 36.0%, respectively. Over multiple iterations of random samples, Social Spring reduced errors by 10% when simulating encounters of radius 3 m and 30% when simulating encounters of radius 5 m. Note, of course, that these results will vary for both different environments and different underlying localization schemes.

Social Spring is capable of reducing indoor path estimation errors for a given, underlying localization scheme by incorporating real-time, relative distance encounters between users. Results from simulated user trajectories and dead reckoning as well as real data collected from a preliminary study indicate that the proposed architecture can dramatically reduce the average estimation error, with increased effectiveness for localization schemes suffering from poor accuracy. This work demonstrates that GRP methods can be used to greatly improve pre-existing indoor localization schemes, though it does not go so far as to suggest or analyze a method for the pairwise distance measurements. We will expand upon this work through the introduction of

ultra-wideband ranging systems, requiring the development of more intricate models of range errors. This is discussed in Chapter 8

CHAPTER 5

Graph Realization for Fingerprint Modeling

Inertial indoor tracking techniques like those examined in the previous chapter suffer from drift over time and must be combined with absolute positioning information to provide an initial position and angle estimate. On the other hand, in many scenarios analytical range- and angle-based methods are impractical due to the complexities and nonlinearities of RF propagation [RM09]. Because of this, many of the more accurate indoor positioning techniques rely on training models of signal propagation over large spaces and using the distinct patterns and variations captured by those models to predict, often in a probabilistic manner, a user's position across time.

Though inertial-based tracking techniques suffer from drift and provide only relative position and angle information, they can provide accurate trajectory estimates over short time periods. This section describes a method by which pedestrian dead-reckoning (PDR) may be used to build a model for Gaussian-distributed RF signal propagation with minimal user effort. Specifically, we model the spatial variation of an RF signal originating from a stationary anchor node as a Gaussian process ($\mathcal{GP}$) whose spatial distribution is estimated by noisy samples provided by a pedestrian dead-reckoning filter based on a foot-mounted inertial measurement unit (IMU) [Fox05]. This is illustrated in Fig. 5.1. Additionally, the PDR filter, having no notion of absolute position or angle, must be periodically corrected as the user walks through known, fixed positions called landmarks. The periodic correction of the PDR estimates is achieved through a *graph realization* using a nonlinear least squares solver, demonstrating improved results over previous drift correction techniques.

By fitting a Gaussian process to these sparse, noisy samples, the training time and effort normally required to provide accurate fingerprinting models can be reduced. In addition, this

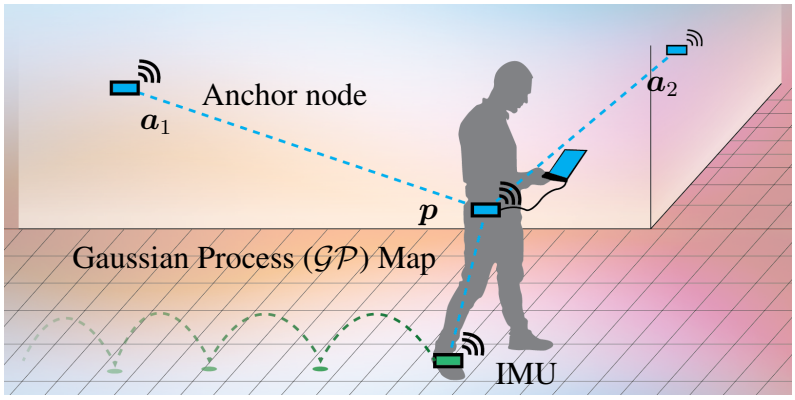


Figure 5.1: Fixed anchor nodes transmit RF signals sampled by a receiver p whose position is estimated by IMU-based dead reckoning.

approach is complementary to those used in the literature of simultaneous localization and mapping (SLAM), as in [AR12, FFL07], in that it can be used to improve the inertial estimation accuracy and thereby reduce the mapping convergence time. We evaluate the proposed training process for a 2500 m^2 multi-room indoor venue and analyze the effect of drift and position correction frequency for several test cases. In short, the contributions of this section are as follows:

- We introduce a lightweight, low-overhead training mechanism for generating Gaussian process maps for subsequent use in indoor positioning systems.
- We evaluate our generated Gaussian process fingerprint maps over a 2500 m^2 multi-room indoor venue for a given localization scheme.
- We further analyze the effect of inertial drift and manual positional correction frequency on the proposed Gaussian process training scheme.

5.1 Methodology

Consider a user carrying a mobile device with an unknown position $\boldsymbol{p}(t) = [x(t)\ y(t)\ z(t)]$ traversing an indoor space containing k fixed anchor nodes with known locations $\boldsymbol{a}_i \in \mathbb{R}^3$ for $i = \{1, \ldots, k\}$. As the user moves about, the RF signals observed at $\boldsymbol{p}(t)$ originating from

$\boldsymbol{a}_i$ will evolve as a function of the distance $\|\boldsymbol{p}(t) - \boldsymbol{a}_i\|_2$ as well as multipath, fading, and other nonlinear propagation effects. If we assume that the height $z(t)$ of the mobile device is relatively constant, the RF measurements sampled by the mobile receiver (e.g. signal strength or time-of-flight) at a given point originating from $\boldsymbol{a}_i$ can be accurately modeled as a Gaussian random variable $\mathcal{N}_i(\mu_{x,y}, \sigma^2_{x,y})$. Additionally, the parameters $\mu_{x,y}$ and variance $\sigma^2_{x,y}$ themselves evolve as multivariate Gaussian random variables over the $\mathbb{R}^2$ plane, described by the Gaussian process $\mathcal{GP}_i$ whose hyperparameters serve as an estimate of the underlying Gaussian random variable at each point $[x, y]$ [YW13]. Each process $\mathcal{GP}_{i=1,\ldots,k}$ for each anchor node $\boldsymbol{a}_i$ and each signal of interest can be trained using Maximum Likelihood Estimation given adequate samples spread over the region of interest (ROI).

Gathering the RF measurements and corresponding true positions required to train each $\mathcal{GP}$ can be quite time-consuming and burdensome—this is oftentimes cited as the main drawback to fingerprint-based positioning schemes. However if the user carries with her an inertial measurement unit (IMU) in addition to the wireless receiver, annotating collected measurements with position data can be largely automated by intelligently integrating PDR estimates with periodic position corrections from *landmarks* with known positions.

5.1.1 Inertial Drift Correction

Pedestrian dead reckoning provides an estimated trajectory $P_0 \in \mathbb{R}^{n\times 3} = [\boldsymbol{p}_0(t_0), \boldsymbol{p}_0(t_1), \ldots, \boldsymbol{p}_0(t_n)]^T$ with accurate *relative* distances and angles over short time periods, but suffering from considerable inertial drift over large periods of time. In order to leverage PDR estimates to automatically annotate the RF measurements collected during $\mathcal{GP}$ training, we must first perform drift correction on P_0 through periodic position corrections as the user walks over a landmark of known position. As a user passes through known landmarks at given times described by time set $T_{lm} \subseteq \{t_0, \ldots, t_n\}$ and position matrix $P_{lm} = [\boldsymbol{p}_{lm}(t_{lm} \in T_{lm})]$, the PDR trajectory estimation is segmented into paths beginning and ending with known positions. These known positions can then be used to retroactively correct the initial estimates P_0 provided by dead-reckoning. Prior methods of performing this correction include linear drift correction [CTS07b], where the

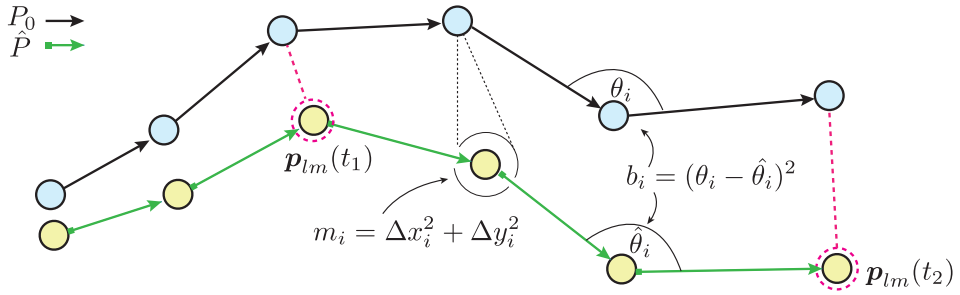


Figure 5.2: Drift correction residuals, including relative motion constraints m_i, angle bias constraints b_i, and landmark constraints.

difference in the predicted and corrected end positions is distributed evenly over all intermediate points, and radial drift correction [ST12], where drift is modeled as both a rotation and a scaling. These methods treat the inertial drift at each time segment independently, yielding computationally efficient algorithms at the cost of a sub-optimal drift compensation scheme. We propose a nonlinear least-squares (NLLS) method that estimates the time-evolving drift of the entire inertial estimate jointly given the set of landmark encounters P_{lm}. Specifically, we propose a drift correction scheme that seeks a solution to the problem

$$\hat{P} = \arg \min_{\hat{\boldsymbol{p}}_{i=1,\dots,n}} \sum_{i=2}^{n} w_m \|m(\boldsymbol{p}_{i-1}, \boldsymbol{p}_i)\|^2 + w_b \|b(\boldsymbol{p}_{i-1}, \boldsymbol{p}_i)\|^2 + \sum_{t \in T_{lm}} w_{lm} \|\boldsymbol{p}_{lm}(t) - \boldsymbol{p}_0(t)\|_2 \tag{5.1}$$

Where the function $m(\boldsymbol{p}_{i-1}, \boldsymbol{p}_i)$ describes the relative motion from $\boldsymbol{p}_{i-1}$ to $\boldsymbol{p}_i$, preserving the original PDR-estimated path structure with weight w_m, and the function $b(\boldsymbol{p}_{i-1}, \boldsymbol{p}_i)$ describes the relative angle bias (offset) from $\boldsymbol{p}_{i-1}$ to $\boldsymbol{p}_i$, allowing for angular gyroscopic drift with penalty weight w_b. Finally, w_{lm} is set high enough to force point $\hat{\boldsymbol{p}}(t)$ to coincide with landmark position $\boldsymbol{p}_{lm}(t)$. By selecting w_m to be the reciprocal of motion estimation variance over short intervals and w_b the reciprocal of variance due to long-term inertial drift (due to drift in gyroscope integration), Eq. (5.1) provides an accurate reconstruction $\hat{P}$ of the true path, say P^*. These residual functions are illustrated in Fig. 5.2, where the initial path estimate is shown by blue dots connected by black arrows and the drift-corrected path is yellow connected by green, block-terminated arrows.

5.1.2 Generating Gaussian Process Maps

Measurements gathered at each position $\boldsymbol{p}(t)$ are denoted $z_i(t)$ corresponding to time t and anchor a_i. Each measurement can be modeled as a perturbation on top of an analytical model $f_i(\boldsymbol{a}_i, \boldsymbol{p}(t))$ describing the expected measurement of that signal at $\boldsymbol{p}(t)$. The measurement can then be written as $z_i(t) = f_i(\boldsymbol{a}_i, \boldsymbol{p}(t)) + \epsilon^i_{x,y}$, where $\epsilon^i_{x,y} \sim \mathcal{N}(\mu^i_{x,y}, \sigma^2_{x,y})$ is the Gaussian-distributed perturbation. Estimating the Gaussian process describing the evolution of each $\epsilon^i_{x,y}$ across space requires estimating the covariance matrix across all function values f_i parameterized by $\{x, y\} \in ROI$ [MST15]. In order to estimate this covariance matrix accurately and thus generate an accurate $\mathcal{GP}$, the positions $\boldsymbol{p}_i \in P^*$ must provide an adequate cover of ROI, and similarly $\hat{P}$ must accurately approximate P^*.

5.1.3 $\mathcal{GP}$-Based Indoor Positioning

The Gaussian process maps discussed above describe the stochastic nature of an RF signal across space. This allows for subsequent position estimation given a time-series of measurements $z_i(t)$. Specifically, given a single measurement $z_i(t)$, the likelihood L that the measurement was taken at position $\boldsymbol{p} = [x, y]$ is given by the normal density function: $L = \frac{1}{2\pi\sigma^2_{x,y}} e^{-(z_i(t)-\mu_{x,y})/2\sigma^2_{x,y}}$. Similarly, the error residual r given a candidate position $\hat{\boldsymbol{p}} = [\hat{x}, \hat{y}]$ can be thought of as $r = [z_i(t) - \mathcal{GP}_\mu(\hat{x}, \hat{y})]/\mathcal{GP}_{\sigma^2}(\hat{x}, \hat{y})$, where $\mathcal{GP}_\mu(x, y)$ and $\mathcal{GP}_{\sigma^2}(x, y)$ are the Gaussian process mean and variance respectively at position $[x, y]$. The position can then be estimated by finding the position that maximizes the likelihood L as in Dynamic Bayesian Network variants like particle filters [GM13] or by finding a position that minimizes the square residual r^2 as in a least squares solver. The accuracy of either technique depends largely on the accuracy of the underlying $\mathcal{GP}$ maps and consequently on the ability to estimate inertial drift correction in the $\mathcal{GP}$ training phase. In Section 5.2 we present results describing a nonlinear least-squares position estimation algorithm adopted from [MST15] that fuses inertial information with r^2 minimization.

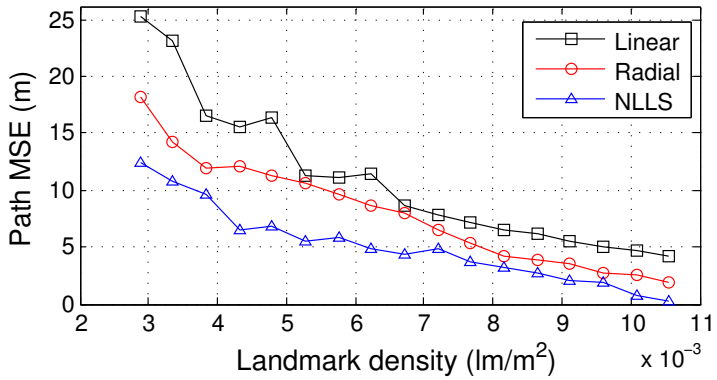


Figure 5.3: Drift correction accuracy vs. landmark density

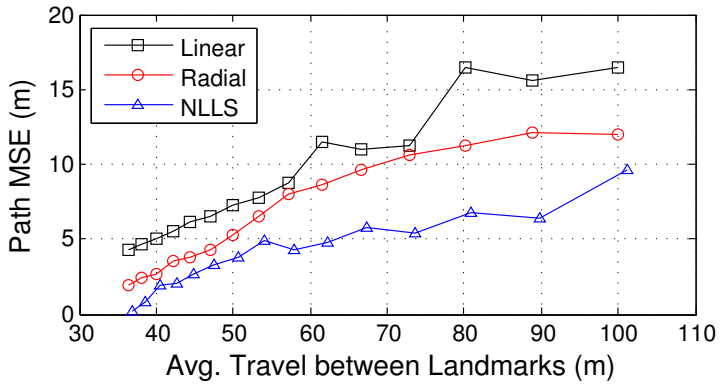


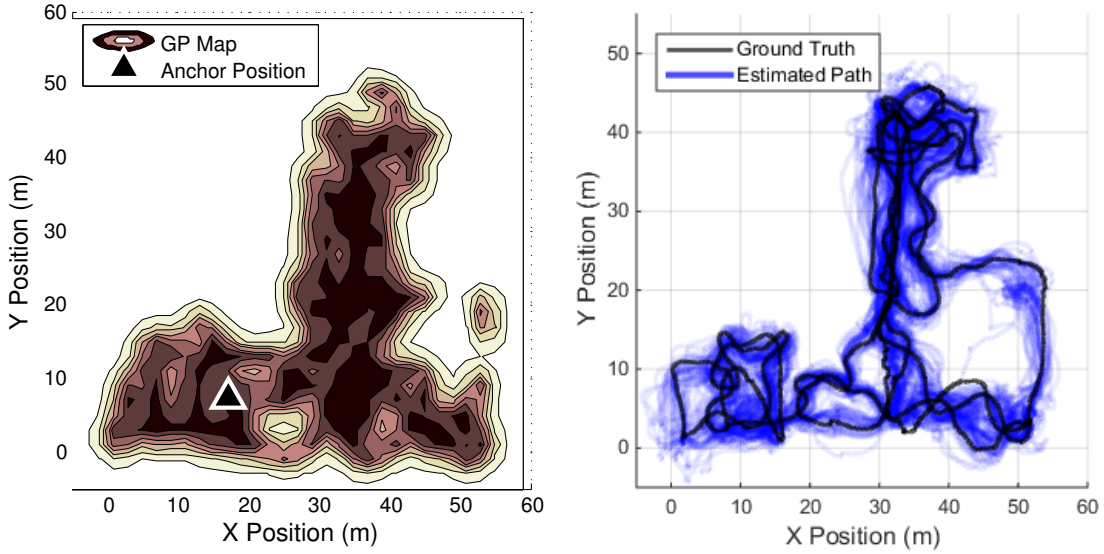
Figure 5.4: Drift correction accuracy vs. landmark distances.

5.2 Evaluation

We evaluated our inertial drift correction scheme, $\mathcal{GP}$ training, and $\mathcal{GP}$-based position estimation algorithm in a large, 2500 m^2 multi-room venue using 9 anchor nodes for both time-of-flight and signal strength measurements. In addition, we generated a $\mathcal{GP}$ map of magnetic field strength to further improve estimation accuracy and demonstrate how the proposed methods can be generalized to arbitrary signals, provided that they are normally distributed across space.

5.2.1 Landmark Drift Correction

Manual landmark correction increases drift correction accuracy at the burden and cost of more laborious training. We seek to minimize the number of landmarks and thus training overhead through the joint inertial drift compensation scheme discussed in Section 5.1.1. We performed



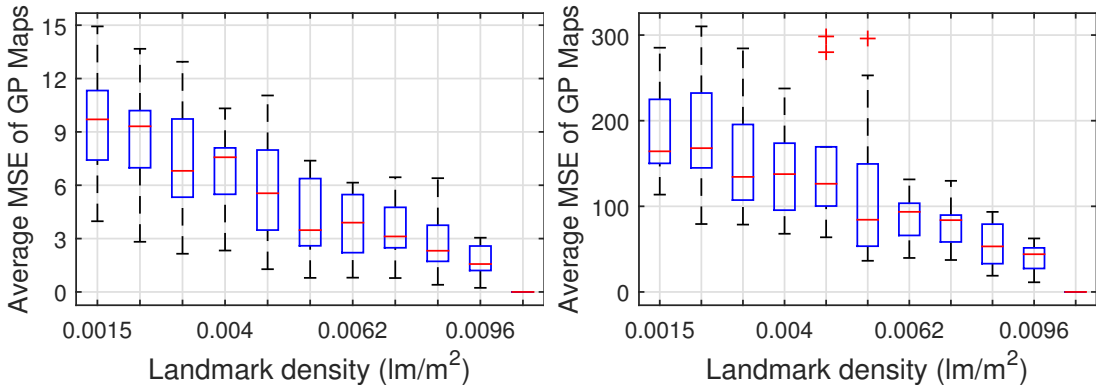
(a) $\mathcal{GP}$ map (b) Estimated paths

Figure 5.5: Example $\mathcal{GP}$ map for time-of-flight from one anchor node (a) and the estimated patths from one 15 minute walking period and various landmark numbers (b).

this compensation using linear, radial, and the proposed NLLS scheme discussed above over several 15 minute periods of inertial data. The resulting path accuracies are shown in Figs. 5.3 and 5.4, where drift correction improves both with increased landmark density and reduced travel distance between landmarks. The proposed NLLS drift correction provides better accuracy than both linear and radial drift correction and can do so with fewer landmarks and with greater distances between landmarks, greatly reducing training overhead. Specifically, with fewer than one landmark per 100 m^2, we can achieve accurate path estimation with less than 1 m MSE. In other words, if we manually annotate a single point in a 100 m^2 region, we can treat PDR estimates as ground truth measurements to within 1 m error.

5.2.2 Convergence of $\mathcal{GP}$ Maps

Accurate $\mathcal{GP}$ maps are required to capture the complexities of signal propagation in a given space. Fig. 5.5a shows one example of a $\mathcal{GP}$ map for time-of-flight measurements from a single anchor, exhibiting highly nonlinear deviations from the ideal analytical model depicted by lighter coloring. The increasing accuracy of drift compensation due to additional landmarks



(a) $\mathcal{GP}$ mean convergence (b) $\mathcal{GP}$ variance convergence

Figure 5.6: $\mathcal{GP}$ mean squared error convergence for a 2500 m^2 room vs. landmark density.

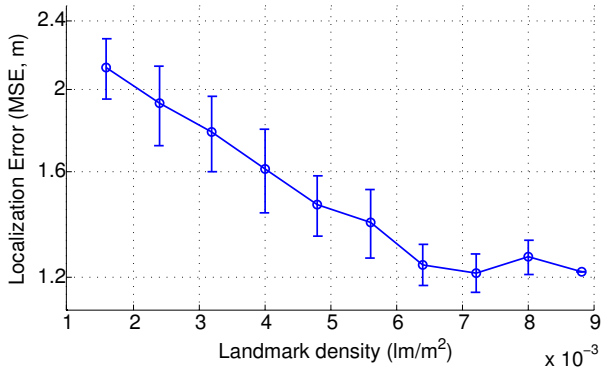


Figure 5.7: $\mathcal{GP}$-based NLLS Localization errors with 95% confidence intervals for a 2500 m^2 room as a function of training landmark density.

causes a corresponding increase in the accuracy of the generated $\mathcal{GP}$ maps to a point, after which a more accurate $\mathcal{GP}$ map will yield only marginal improvements for position estimation results. For two Gaussian process maps $\mathcal{GP}_i$ and $\mathcal{GP}_j$, we define the distance between the two maps as the average squared distance at each position $[x, y]$ in the region of interest—that is, $\frac{1}{|ROI|}\sum_{(x,y)\in ROI} \|\mathcal{GP}_i(x,y) - \mathcal{GP}_j(x,y)\|^2$. This distance can be calculated for all hyperparameters describing the $\mathcal{GP}$—namely mean and variance in this case. The evolution of these errors as a function of landmark density is shown in Fig. 5.6. This figure shows that the $\mathcal{GP}$ maps converge at a steady rate, reaching low average errors around a density of 0.005 landmarks per m^2, or roughly 1 landmark for every 200 m^2.

5.2.3 Position Estimation

To give more intuition into the accuracy of the $\mathcal{GP}$ maps discussed above, we performed position estimation tests across a range of landmark densities. For each result, we generated $\mathcal{GP}$ maps based on RF time-of-flight, signal strength, and magnetic field strength. Each $\mathcal{GP}$ is generated using inertial estimates corrected with only the number of landmarks specified. The result of one 15 minute test is shown in Fig. 5.5b, where the estimation results from a range of landmarks are overlaid such that more transparent colors represent fewer landmarks. Fig. 5.7 shows that for the 2500 m^2 venue very few landmarks (3 in the entire venue) are required to achieve a localization accuracy of around 2 m. Additionally, position estimation accuracy improves to below 1.5 m around a density of 0.005 landmarks per m^2, or around 1 landmark per 200 m^2. This corresponds to the point at which the median error in the $\mathcal{GP}$ maps has begun to converge, as shown in Fig. 5.6.

5.3 Discussion

We have presented an improved method of inertial dead reckoning drift compensation as well as Gaussian-process based RF signal training methods with the goal of reducing training overhead for fingerprint-based indoor positioning and navigation systems. Our evaluations on a 2500 m^2 multi-room venue demonstrate improved drift correction over several other techniques and accurate position estimation even while reducing drift correction landmarks to a density of 1 per each $200\ m^2$ area. We believe this approach can aid in reducing training time and overhead for a number of localization schemes in addition to those addressed in this work. In particular, the results presented herein demonstrate that fingerprint-based indoor positioning approaches need not necessarily be overlooked due to the effort required in training the required models—for small to medium-sized spaces, these models can be generated with very little effort if done so carefully. Finally, fusion of absolute (landmark) and relative (inertial) constraints using a graph realization solver serves to improve path reconstruction accuracy when compared to traditional drift correction approaches, once more underscoring the efficacy of GRP approaches in mobile

device localization.

CHAPTER 6

Pathfinding by Joint Optimization over Pressure Data

In the previous chapters we looked at using relative distance and relative motion constraints as added edges in a graph realization problem. In this chapter, we look at a third source of relative constraints—pressure collected from mobile barometer sensors. Interestingly, these sensors are found on an increasing number of smart devices. Though seemingly innocuous, these sensors can be combined and analyzed to make surprising inferences. In this work, we demonstrate how a mobile application can use pressure data collected from a device's barometer in order to detect with high accuracy likely paths along which the user has recently traveled. We further analyze the ability to predict unknown mobile trajectories in terms of the variance in barometer pressure and geographical elevation, demonstrating cases in which more than 70% of paths can be accurately predicted. In doing so, we do not use a graph realization solver in the traditional sense, though the problem of predicting a driving path is enabled in large part by structuring potential road segments and intersections as edges and nodes in a graph, respectively. We then make use of dynamic programming approaches and intelligent pruning heuristics to detect a realistic sub-graph within a larger graph that best represents the measured data.

6.1 Background

In the torrent of new and ever-improving mobile computing devices that has become the technical and social norm of newer generations, developers are continually searching for methods, whether through hardware or software, to provide their users with new and exciting capabilities. These new capabilities often include new sensing modalities or intelligent ways of fusing pre-existing sensors to sense or infer some physical phenomenon or stimulus.

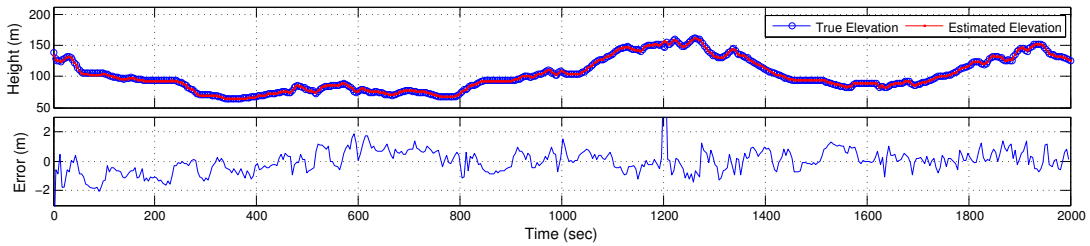


Figure 6.1: Elevation estimation from pressure with corresponding error, using simple linear model.

Recent mobile devices have introduced yet another sensing modality in the form of barometric pressure sensors. These sensors can already be seen on mobile devices such as the Apple iPhone 6, Google Nexus 5, and Nexus 6. In addition, both Apple's iOS and Google's Android OS treat the barometer as a non-private sensor such as an accelerometer, or gyroscope. In other words, an application that wishes to read a mobile device's barometer can do so without alerting the device user. In addition, the barometer introduces a level of geographical dependency unseen in previously available 'public' sensors, including the magnetometer. While a magnetic compass might behave differently in different geographical locations, a barometric sensor is *designed* to differentiate between different pressures and thus, to a large degree, different elevations. In this chapter, we show that the high correlation between pressure sensors and elevation introduce the potential for estimating driving paths for a given user, on the one hand demonstrating a privacy and security concern but on the other motivating the need for methods of fusing multiple sensory data to pre-existing localization methods.

More specifically, this chapter demonstrates that pressure data collected while a user is driving can be used to determine with high probability the paths along which that user has driven. To begin, a user's coarse-grained location estimate is obtained by associating the device's IP Address with an ISP's geolocation—a procedure that can be done by a remote server without device permission. From there, the user's time-series pressure data is compared against a database of possible road segments and their corresponding elevation signatures. Under certain conditions regarding the uniqueness of the observed pressure data with respect to the elevation of the underlying map, this allows us to obtain a series of 'ranked' paths, ordered by descending likelihood.

Though many factors affect the pressure reported by barometer sensors (not the least of which are weather, air movement, and sensor drift), mobile barometric sensors can still be correlated with elevation *changes* with surprising precision. Using a simple linear model (the details of which are discussed in Section 6.2), height can be predicted to within an error of several meters. An example of this correlation is shown for 30 minutes of driving data in Figure 6.1, where the error rarely exceeds ± 2 m.

6.1.1 Contributions

As a result of the high sensitivity of mobile barometric pressure sensors, it is feasible to correlate a user's motion over a varying landscape to her corresponding pressure data. This correlation, however, is made difficult due to several important factors: first, the conversion from pressure to elevation is time-varying and unknown *a priori*. Second, the user's vehicle is traveling at an unknown speed, essentially 'sampling' elevation points at variable and unknown rates. This makes it difficult to directly correlate pressure data to the elevation of a given road segment. Third and most importantly, the search space of possible paths against which to compare the user's collected pressure data is immense, even given coarse grained location estimates such as those obtained from IP Address geolocation.

In order to elucidate the degree to which pressure can be used to determine a user's traveled path in the face of the difficulties mentioned above, we make the following contributions:

- We describe and develop two algorithms for estimating driving paths given pressure data. The first uses greedy heuristics along with dynamic time warping to iteratively estimate and prune likely paths through independent explorations of road graphs. The second uses dynamic programming and dynamic time warping to find a jointly minimal cost path through a graph of road segments. We further evaluate both algorithms in terms of their computational complexities and estimation accuracies.

- We evaluate the performance of our path estimation algorithms over a number of real test cases totaling 150 km and 4.6 hours of driving data.

- By modeling errors in barometer sensors and elevation estimation, we simulate path estimation results for random driving paths selected across a large number of cities with varying geographical landscapes.
- We evaluate the accuracy of our path prediction algorithms in terms of the distinctness of the pressure data with respect to the surrounding landscape, offering insights into the conditions under which driving paths can be accurately predicted.

Related work in this area can roughly be divided into three categories: demonstrations of nefarious or otherwise malicious inferences made from mobile sensory data (excluding location inferences), location- or transportation- specific inference mechanisms using non-private sensor data, and characterizing and mitigating mobile privacy leaks in general

Malicious Mobile Inferences: The body of research demonstrating malicious inferences is ever-growing. Below we provide an abbreviated snapshot of the state-of-the-art. Among the more infamous inferences made from mobile sensory data are those that infer user typing behavior from accelerometer data, such as those described in [MVB12, OHD12, MVC11, ASB12]. The accelerometer has also been demonstrated to leak private information in the form of activity recognition [BI04]. Additional work has demonstrated scenarios when seemingly harmless mobile audio recordings can also leak sensitive, private information. For example, just as mobile touch-typing can be detected from accelerometers, [AA04] demonstrates that touch-typing can also be inferred through audio channels. Further work demonstrates that mobile audio data can even be used to extract RSA keys [GST13] or infer human stress levels [LFR12, CC11, PRH11].

Inferring Location and Transportation: In the realm of location discovery techniques, recent work has demonstrated trajectory identification through inertial navigation (dead-reckoning) for both pedestrians [PPC12] and vehicles [GPL10]. Han *et al.* further demonstrate in [HON12] that accelerometer time-series data can be used to constrain a user's driving path to a subset of possible candidate paths within a given map. Hemminki *et al.* [HNT13] demonstrate methods for inferring transportation modes using mobile accelerometer data, and Sankaran *et al.* demon-

strates methods for inferring transportation modes using barometer data [SZG14]. Finally, Zhou *et al.* demonstrate in [ZDH13] that app usage statistics, network address-resolution, and speaking detection can be used to infer user identity, coarse geo-location, and even whether or not a person has a certain disease. The methods presented in this chapter demonstrate how pressure data collected from mobile barometers can be used to predict driving paths. Unlike methods like those presented in [GPL10] and [HON12], the proposed methods allow for accurate *absolute* path predictions, benefiting from the high correlation between barometer and elevation, as detailed in [MKM14].

Privacy Preservation: Inspired by recent work like those described in the paragraphs above, considerable research effort has focused on providing mechanisms by which to track or mitigate privacy leakage in mobile applications. These efforts include crowdsourcing techniques [AH13], policy-based approaches [BKO10, JMV11, NKZ10], sandboxing approaches [LWG13, BRS11], information tracking approaches [EGC10], and information theoretic analyses [ips]. Given the magnitude of research efforts concerning privatizing and securing mobile sensory data, it is clear that this is an important and pressing matter. The work presented here serves to further underscore the urgency of mobile privacy research, reinforcing the need for privacy-centric design paradigms in future mobile technologies.

6.2 Estimating Elevation

Though barometric sensors are strong indicators of geographic elevation, they are sensitive to a host of other pressure changes as well, making the path from pressure to elevation a non-trivial conversion.

As elevation changes, changes in air density due to Earth's gravitational pull and many other factors cause a pressure gradient dictated by the *barometric formula*. Ignoring the effects of temperature change as a function of altitude, this formula is given in [usa76] as

$$P = P_b \cdot \exp\left[\frac{-gM(h - h_b)}{RT_b}\right] \tag{6.1}$$

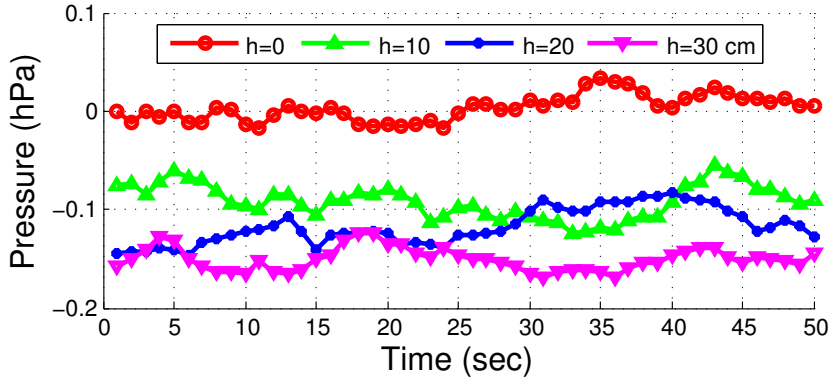


Figure 6.2: Barometer readings from a Nexus 5 at varying height offsets.

where P is the pressure in hecto-pascals (hPa) at height h meters above sea level. g is the acceleration due to gravity, M is the molar mass of Earth's air, and R is the universal gas constant for air. P_b is the pressure measured at the baseline point at height h_b meters above sea level with reference temperature T_b, which all can be measured beforehand. Given variables P and h, we can derive the general equation $h = \alpha + \beta \log(\gamma P)$. Over small changes in elevation (hundreds of meters), the corresponding change in h can be estimated by pressure using a linear model as in

$$h \approx \alpha(t) + \beta P \tag{6.2}$$

for some scalar β and time-varying offset $\alpha(t)$. Over short periods of time (less than 1 hour), this prediction can be quite accurate. In fact, though the Bosch BMP280 pressure sensor used in both the iPhone 6 and Nexus 5 specifies a vertical pressure resolution of about 1 m [bmp], we have experimentally validated relative pressure sensitivity closer to 10s of centimeters, as shown in Figure 6.2. Despite this strong correlation, determining the parameters $\alpha(t)$ and β can be challenging, especially given the time-varying aspect of the offset $\alpha(t)$ due to, among other things, weather.

6.2.1 Elevation Model Estimation

The model parameters $\alpha(t)$ and β in (6.2) dictate the accuracy of *absolute* elevation prediction. Thankfully, the scaling term β can be considered constant over very large ranges in elevation,

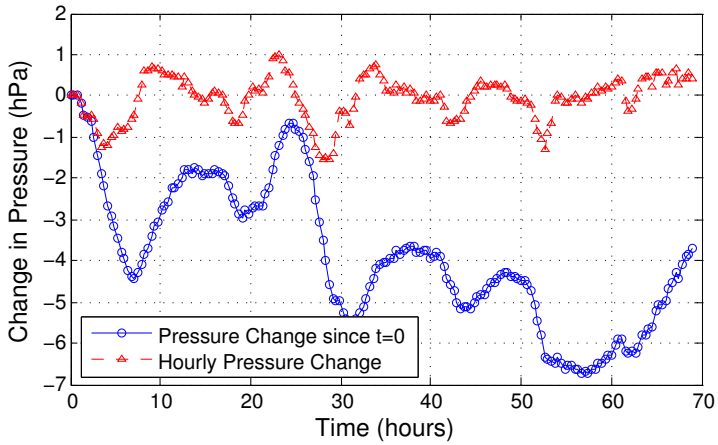


Figure 6.3: Example of pressure changes across multiple days for a constant location.

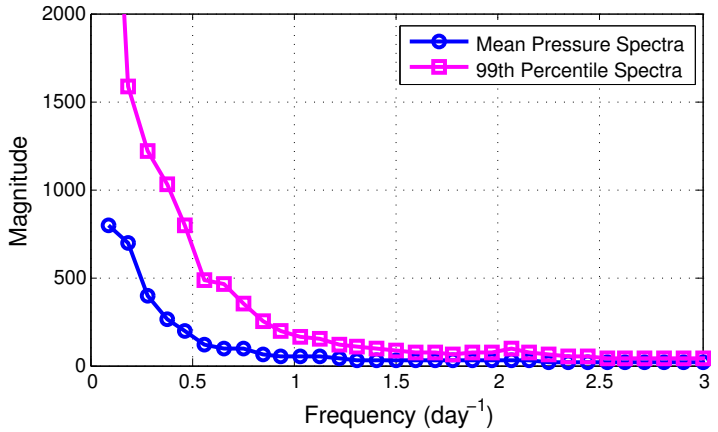


Figure 6.4: Pressure spectra from 2,309 U.S. cities provided by NOAA.

due to the relative flatness of earth's surface with respect to its diameter. The offset $\alpha(t)$, on the other hand, varies wildly with time and coarse location. This can be seen in Figure 6.3, where pressure collected at a static location over a 70-hour period exhibits pressure changes nearing 7 hPa (about 50 m estimation error!). Muralidharan *et al.* describe this problem in detail in [MKM14]. Note, however, that the frequency of this variation is quite low. If we look at the pressure change over 1 hour periods, the change rarely exceeds ± 1 hPa (or roughly 8.3 m). Furthermore, a survey of hourly pressure data from 2,309 cities in the U.S. provided by the National Oceanic and Atmospheric Administration (NOAA) Climatic Data Center [noa] shows that, over a 1 hour period, the average change in pressure for a given weather station is 0.0016 hPa while the 99^{th} percentile of changes is below 1 hPa. Additionally, analyzing these

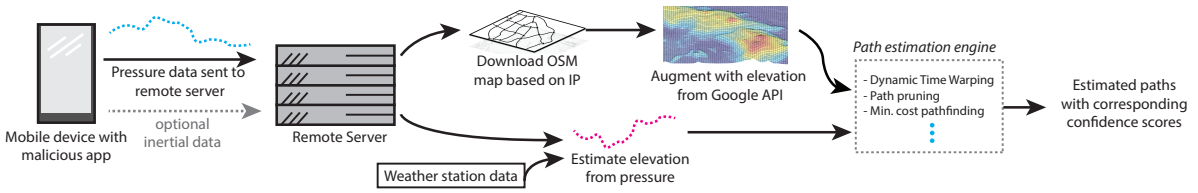


Figure 6.5: System overview, from malicious data collection to path estimation and confidence score.

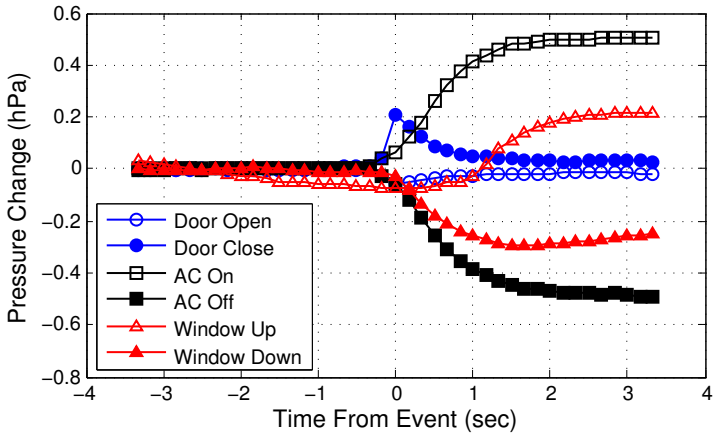


Figure 6.6: Pressure changes for various driving events

changes in the frequency domain indicates that the vast majority of pressure changes happen at the scale of 1 or more days, rather than hourly. This is shown in Figure 6.4. Finally, given the slow dynamics of weather, the very pressure data provided by weather stations can be used to calibrate the offset term $\alpha(t)$ to within 1 hPa error—the typical resolution of pressure reported by weather stations. If such a station does not exist in close proximity to a user's coarse location, in some scenarios *relative* elevation can still be used to estimate a user's driving path with some reduction in estimation accuracy. This is discussed in more detail in Section 6.3.

6.2.2 Pressure Events & Noise Sources

In addition to model dynamics caused by weather, mobile barometers experience 'noise' from a number of sources including those caused by dynamics of air flow. While driving, events that cause notable pressure noise include closing doors, changing the air conditioning (AC), and opening/closing windows. The effects these events have on pressure are shown in Figure

6.6, where events occur at time $t = 0$ and pressure prior to an event is normalized to 0 hPa. The largest magnitude pressure change occurs when air conditioning is fully turned on or off, resulting in a 0.5 hPa change (roughly 4 m estimation error). While for the most part a user will not change AC from fully on to fully off or vice-versa, Figure 6.6 underscores the need for path detection algorithms that remain robust to slight perturbations in errors, including constant offsets.

Finally, barometer sensors themselves are not perfect and typically exhibit (small) drift over time. This error is in general non-gaussian, exhibiting temporary drifts from the true pressure while periodically returning to accurate estimates. We propose modeling this error using an Ohnstein-Uhlenbeck diffusion process, which is similar in flavor to a low-pass-filtered white noise process [Enr08]. This error model will be used to generate realistic barometric pressure traces for simulations, as outlined in Section 6.4.2.

6.3 System Overview

Our path prediction system is composed of two main components: (1) a malicious mobile app that continually monitors barometer data (and optionally accelerometer and gyroscope data) without user consent, periodically sending the data back to the second component: (2) a centralized server that maintains road maps and elevation data and uses the collected sensory data together with the map and elevation database to estimate likely paths. This is shown in more detail in Figure 6.5.

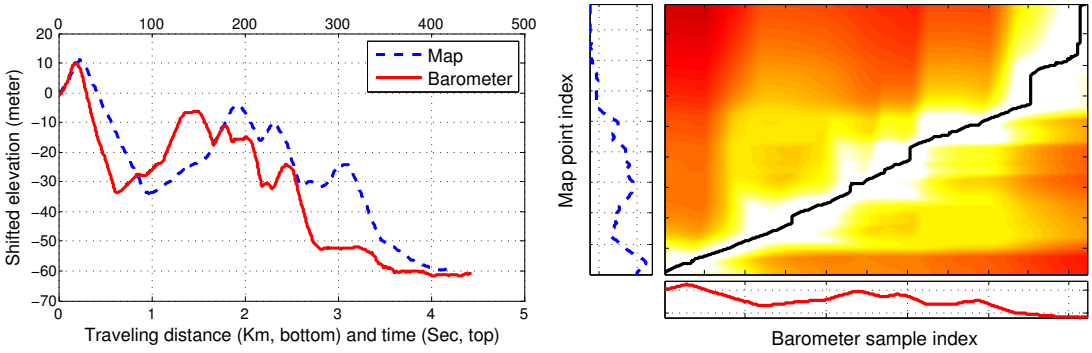
Upon contact with a remote app, the server downloads the corresponding road maps and elevation data from publicly available online databases based on the rough location from the communicating app's IP Address. Specifically, the server downloads and manipulates data from (i) Open Street Map (OSM) [Ope15], providing road topology including segments and intersections, and (ii) the Google Elevation API [Goo15], which provides a database from which to query the elevation of individual latitude and longitude points. Upon receiving the barometer data from the mobile app, the elevation conversion module converts the pressure readings to an

estimate of absolute elevation values. If the sensor data is collected in a region close to a weather station, the pressure from that station can be used to train the offset term $\alpha(t)$ in (6.2) (the β term here remains more or less constant across time and location). If a nearby weather station is not available, a less accurate estimation can be made and the system can (a) iterate over possible values of α or (b) reduce to comparing *relative* changes in elevation instead of *absolute* pressure. These options will be discussed in more detail in subsequent sections. Finally, we perform a number of pattern recognition routines including dynamic time warping (DTW) to perform path matching. These estimation routines, encompassed in two path estimation engines, is the main contribution of this work and the subject of Sections 6.3.3.1 and 6.3.3.2.

6.3.1 Elevation Map Generation

In order to estimate the path along which a user has driven, the server must first generate a database of possible road segments and their corresponding elevations. To do this, we combine the OSM road topologies with Google's publicly available Elevation API. Road maps are downloaded from OSM in an XML format composed of 64-bit unique *node* identifiers and their corresponding latitude and longitude values. These 'nodes' are connected by elements termed *ways*—ordered lists of connected road nodes. In general, either endpoint of a *way* aligns with road intersections. Ways composed of more than 2 nodes are often used to better represent road curvatures between intersections.

From the OSM road nodes and ways, we construct a graph $G = (V, E)$ where V is the set of road nodes and E is the corresponding edge set described by the OSM *way* elements. OSM also uses 'ways' to represent landmarks, building countours, etc. As a consequence, G may be composed of several disconnected sub-graphs. To cope with this, the map is preprocessed in order to filter out any connected component in G smaller than a predefined size. In addition, we add extra internal nodes in each way at intervals of roughly 10 meters, extending the original graph to $G' = (V, E, N)$, where $n_i \in N$ are all intermediate nodes and V represents only intersection nodes. Each edge $e_{a,b} \in E$ is defined by the end-point intersection nodes $\langle v_a, v_b \rangle$ and their corresponding *way* of intermediate nodes, $[n_a, n_1, n_2, \ldots, n_b]$. Additionally, all nodes



(a) Candidate path vs. barometer-based elevation.

(b) Corresponding DP matrix D with min-cost path.

Figure 6.7: An illustration of path elevation matching using dynamic time warping.

$n \in N$ are augmented with latitude, longitude, and elevation attributes.

6.3.2 Path Matching

Armed with a database of road segments and corresponding elevations, the goal of the server is to determine which path $\hat{p} = [v_0, v_1, \ldots, v_k]$ through graph G' is most probable given the observed pressure time series $\mathbf{s}^b$. This is made particularly difficult given the large number of possible paths through G' (which is infinite, if loops are allowed), and the unknown speed of the user. Because of the unknown user speed, the sampled pressure data behaves as a noisy sampling of elevation through the true path p^* with variable sampling rate. To combat this, we adopt a dynamic time warping approach as described below.

6.3.2.1 Dynamic Time Warping

Because the user is traveling at an unknown and variable speed, the corresponding pressure data behaves as a sampled version of the true elevation but with variable sampling rate. In other words, the collected pressure may be of a short duration, long duration, or it may contain pauses (when the vehicle is not in motion) or increased speeds. Because of this, we leverage dynamic time warping (DTW) algorithms like those developed for speech recognition (where the same word may be spoken with variable durations) [Vin68]. In DTW, two time-series signals are

compared against each other by means of a cost matrix. If the two series are denoted by column vectors $\mathbf{s}^p = \{s_i^p\} \in \mathbb{R}^{N_p}$, representing the elevation corresponding to a candidate **p**ath in G', and $\mathbf{s}^b = \{s_j^b\} \in \mathbb{R}^{N_b}$, corresponding to the **b**arometer-based elevation estimate, the cost matrix C contains $N_p \times N_b$ elements where element $c_{i,j} = f(s_i^p, s_j^b)$. The function $f(\cdot)$ serves as a distance function to represent the difference between the two signals at given indices and is typically defined by an ℓ_2-norm. The goal of DTW is to find the minimum-cost path through cost matrix C starting at $c_{0,0}$ and ending at c_{N_p,N_b} and constrained to a set of possible transitions defined in T. A transition is defined as a 2-tuple (Δ_p, Δ_b), meaning that an element $c_{i,j}$ can be reached by $c_{i-\Delta_p,j-\Delta_b}$. A typical transition set is defined as $\{(0,1),(1,1),(1,0)\}$. Since the transition set is finite and backward transitions are disallowed, the optimal path can be determined using a dynamic programming (DP) approach. The details of the DTW algorithm are given in Algorithm 1. For each element $d_{i,j}$ in a DP matrix D, we store the minimum cost of all possible paths from $c_{0,0}$ to $c_{i,j}$. Each element $\psi_{i,j}$ in a traceback matrix Ψ records the last transition which lead to the minimum cost of $d_{i,j}$. Thus, the final similarity between $\mathbf{s}^p$ and $\mathbf{s}^b$ is embedded in d_{N_p,N_b}. If $\mathbf{s}_p$ and $\mathbf{s}_b$ are similar, their corresponding DTW score will be low, and if they are dissimilar their cost will be high, regardless of variable lengths or sampling rates. An example of the DTW procedure is illustrated for example map path- and barometer-based elevation data in Figure 6.7(a), with the corresponding DP matrix D in Figure 6.7(b). Following Algorithm 1, the complexity of the DTW algorithm is $\mathcal{O}(N_p \cdot N_b)$.

6.3.3 Candidate Path Generation

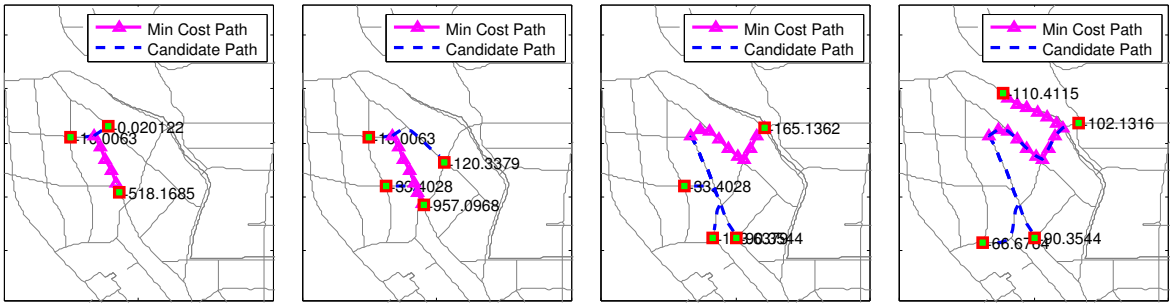
Given a method of matching candidate elevation data to observed pressure data (using DTW), the server must now compare the estimated elevation against a set of possible candidate elevations and their corresponding paths. In other words, the server must search for a path $\hat{p}$ through G' such that the elevation along $\hat{p}$, denoted by $\mathbf{s}^{\hat{p}}$, and and the elevation converted from barometer data, denoted by $\mathbf{s}^b$, are similar, i.e.,

$$\hat{p} = \arg\min_p DTW(\mathbf{s}^p, \mathbf{s}^b)$$

Data: Signals $\mathbf{s}^p \in \mathbb{R}^{N_p}$, $\mathbf{s}^b \in \mathbb{R}^{N_b}$
Result: similarity score R_{score} and traceback vectors $\mathbf{v}^p \in \mathbb{Z}_+^{N_p}$ and $\mathbf{v}^b \in \mathbb{Z}_+^{N_b}$
Cost matrix: $C = \{c_{i,j}\} = |s_i^p - s_j^b|^2$;
DP matrix: $D = \{d_{i,j}\} = 0$;
$d_{0,j} = \infty, \forall j = 1 \dots N_b$;
$d_{i,0} = \infty, \forall i = 1 \dots N_p$;
Traceback matrix: $\Psi = \{\psi_{i,j}\} = \phi$;
for r *in* $1 \dots N_p$ **do**
    for c *in* $1 \dots N_b$ **do**
        $d_{r,c} \leftarrow \min_{t=(\Delta r, \Delta c)} (d_{r-\Delta r, c-\Delta c}) + c_{r,c}$;
        $\psi_{r,c} \leftarrow \arg \min_{t=(\Delta r, \Delta c)} (d_{r-\Delta r, c-\Delta c})$;
    end
end
$R_{score} \leftarrow d_{N_p, N_b}$;
traceback: $r = N_p$, $c = N_b$;
while $r > 1$ *or* $c > 1$ **do**
    $v_r^p \leftarrow c$;
    $v_c^b \leftarrow r$;
    $(\Delta r, \Delta c) \leftarrow \psi_{r,c}$;
    $r \leftarrow r - \Delta r$;
    $c \leftarrow c - \Delta c$;
end

Algorithm 1: Dynamic Time Warping via Dynamic Programming with traceback.

Given the size of the map in which the user is assumed to be located (estimated by IP address geolocation), this search space can be quite large. In fact, if we allow for loops in paths, the search space itself can be infinite—i.e., there are infinite combinations of paths in G'. Because of the massive search space of possible paths, we have developed two path-finding algorithms. The first is a greedy, agent-based approach, where agents are created at each node and continue to traverse the graph in a breadth-first manner. The second approach uses a jointly-optimal dynamic programming approach. While the former scales well to larger map sizes, the latter provides dramatically increased accuracy. We describe both approaches in detail in the following sections.



(a) Greedy pathfinding, $t = 0$. (b) Greedy pathfinding, $t = 1$ (c) Greedy pathfinding, $t = 2$ (d) Greedy pathfinding, $t = 3$

Figure 6.8: Snapshots of an agent from greedy pathfinding, exploring 4 possible paths.

6.3.3.1 Greedy Path Finding

In order to determine which of all possible candidate paths would most likely generate the pressure data collected by a mobile app, we can use an agent-based approach in which each agent is traversing one possible path. Specifically, we consider $|V|$ possible agent starting locations, where each agent is responsible for exploring possible paths whose starting point is $v_i \in V$. To do this, each agent A_i (of which there are $|V|$ initially) performs a breadth-first traversal over V. Because G' is not necessarily (and in general is not) free of cycles, a breadth-first traversal will quickly escalate into an exponential problem. To combat this, each agent ensures that no path it explores creates a loop of length less than a threshold Γ_{loop}. If Γ_{loop} is large enough, this allows for reasonable driving trajectories while greatly limiting the search space.

At each iteration, agents will explore new paths from each of their leaf nodes. After each exploration, pruning occurs in order to limit the computational complexity of the algorithm and thereby improve runtime. Pruning occurs in three stages: (1) after all agents have finished exploring new nodes, the solver calculates all candidate path scores using a modified, greedy version of Algorithm 1 in which the similarity score R_{score} represents the normalized minimum cost between the partial map path p' and *any* index in the barometer-estimated elevation, i.e., $\min_i \left(\frac{d_{|p'|,i}|p'|}{i} \right)$, rather than the entire barometer trace. These scores are sorted and a threshold T_{score} is computed such that T_{score} is the R^{th} percentile of the sorted scores; (2) All agents

are instructed to prune any of their paths whose DTW score exceeds T_{score}; and finally (3) all agents are instructed to prune their worst paths until they contain no more than $\Gamma_{maxpath}$ paths. This allows for diversity in possible path starting locations and reduces the complexity of the search algorithm to polynomial time. The percentile R of paths to keep serves as a knob to trade off runtime and accuracy. A high R removes paths based on a stricter threshold, while a low value of R prunes paths more slowly, preventing premature pruning of paths that may otherwise have performed well in future iterations. Finally, if an agent has pruned all possible paths and no paths are left to explore, the entire agent is removed. At a certain point, paths have been pruned to the point where only the minimum $\Gamma_{minpath}$ paths exist between all agents. At this point, any new paths explored are still subjected to the pruning rate R, but paths are not pruned if doing so would result in fewer than $\Gamma_{minpath}$ paths. The solver terminates if the minimum cost of all candidate paths does not change more than 1% across a period of 10 iterations. Upon terminating, the greedy solver returns the top ranked paths and their corresponding scores. A series of snapshots from the operation of a single agent in the greedy solver is shown in Figure 6.8. Here we have set $\Gamma_{maxpath}$ to 4, so that at each iteration only 4 paths are being considered (labeled by green squares). For each iteration, the minimum cost path is displayed by a string of magenta triangles, while other candidate paths are displayed with a dashed blue line. At each iteration, there are at most $|V| \cdot \Gamma_{maxpath} \cdot d_{out}^{max}$ paths, where d_{out}^{max} is the maximum out-degree of any node in V. As a result, the time complexity can be bounded by $\mathcal{O}(|V|) \cdot \mathcal{O}(DTW) = \mathcal{O}(|V| \cdot N_p \cdot N_b)$. In practice, we truncate the map elevation segment to at most length N_b, giving a final complexity of $\mathcal{O}(|V| \cdot {N_b}^2)$. These calculations are performed over a number of iterations, but by setting a maximum number of iterations (30 in our experiments), the computational complexity remains unchanged.

6.3.3.2 DP-based Path Finding

The greedy search algorithm explained in the previous section is intuitive and, when the pruning ratio is properly set, can search efficiently over large maps. However, there are several drawbacks to this approach: First, since the pruning removes a certain amount of candidate

paths from future exploration, there is a potential for converging to a local minimum cost path. Second, the greedy search heuristics do not consider path timing information, thus making it impossible to reuse intermediate results for later searches. To overcome these drawbacks, we can instead consider an approach inspired by Dijkstra's shortest path search algorithm, whose underlying algorithm is again solved via dynamic programming. We call this the *DP-based* path finding algorithm.

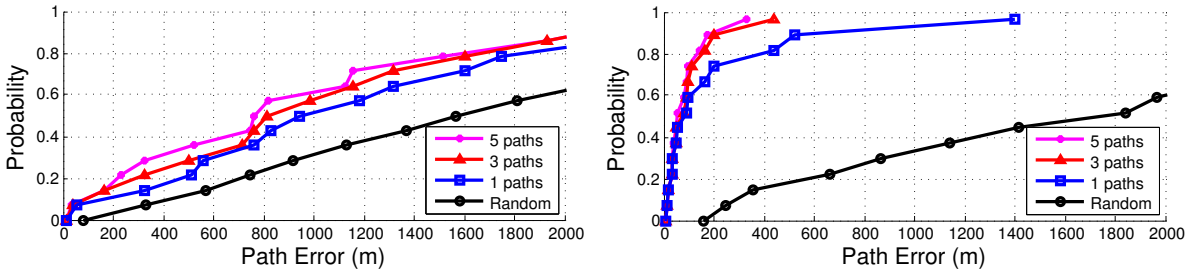
To improve on the greedy path estimation algorithm, we introduce a notion of time—a path is not just a series of locations, but also a series of corresponding timestamps. Thus, we redefine path p to encompass a series of *states* $[q_0, q_1, \ldots, q_k]$ where each state q_i is a 2-tuple (v_i, t_i) indicating that the user has reached intersection v_i at time t_i. Rather than define cost in terms of the DTW score between two entire paths, we can now define the marginal cost in transitioning from state q_i to q_j. Specifically, if we have already discovered a portion of the true path, say $\mathbf{q}' = [(v_0, t_0), (v_1, t_1), \ldots, (v_i, t_i)]$ whose matching cost thus far is δ_{q_i}, then the cost of transitioning to $q_j = (v_j, t_j)$ is defined as $cost(q_i, q_{i+1})$, so that $\delta_{q_{i+1}} = \delta_{q_i} + cost(q_i, q_{i+1})$.

Thus, for each state $q = (v, t)$, the optimal δ_q is defined as a partial path ending at vertex v at time t and can be determined by the following recursive function:

$$\delta_q = \min_{t_- < t, \langle v_-, v\rangle \in E} (\delta_{q_-} + DTW(\mathbf{s}^p, \mathbf{s}^b)) \tag{6.3}$$

$$\delta_{q=(v,0)} = 0, \forall\ v \in V \tag{6.4}$$

where t_- and v_- specify the time and node corresponding to the previous state q_-, $\mathbf{s}^p = Elev(\langle v_-, v\rangle)$ is the elevation along the edge $\langle v_-, v\rangle$, and $\mathbf{s}^b = [s^b_{t_-}, \ldots, s^b_t]$ is the barometer-based elevation estimate from time t_- to t. This recursive definition successfully reduces the exponential number of candidate paths to a polynomial time search algorithm: the complexity is decided by (1) the table size of δ, (2) the number of state transitions, and (3) the complexity of DTW, leading to the final complexity of $\mathcal{O}(|N|^2 \cdot {N_b}^3)$. This does not scale to large maps as well as the greedy approach discussed in Section 6.3.3.1, but the jointly minimal solution provided by dynamic



(a) Greedy solver (b) DP-based solver

Figure 6.9: Path prediction errors for real driving data

programming provides increased accuracy in path prediction.

6.3.3.3 Improving DP-based Search Complexity

The DP-based path finding algorithm suffers from high complexity due to many redundant calculations, both across time and location (i.e., vertex). For example, $\delta_{q=(v,t)}$ is updated by any state $q_{\text{-}}$ with $\langle v_{\text{-}}, v\rangle \in E$. As described in (6.3), DTW is performed for each possible transition to q. It is possible to amortize this DTW cost by flattening out this recursive relation.

The root cause of this redundancy is that traveling from vertex v_a to v_b requires enumerating possible arrival times t_b. If, however, we assume that it takes at least one time unit to travel to any adjacent *intermediate* node $n \in N$ (which implies that the last transition in T described in Sec 6.3.2.1, is removed), the number of possible state transitions reduce to

$$\delta_q = \delta_{q_{\text{-}}} + \min_{\langle n, n_{\text{-}}\rangle} |s_n^p - s_t^b|^2 \tag{6.5}$$

which is bounded by the constant $\mathcal{O}(d_{out}^{max})$, yielding a final complexity of $\mathcal{O}(|N|^2 \cdot N_b)$.

6.3.3.4 Additional Pruning Metrics

In addition to pressure data, a number of other 'public' sensors can be used to further improve path prediction accuracy and prune improbable paths to increase runtime efficiency. Most notably, mobile accelerometers, gyroscopes, and magnetomers can be combined to give accurate

turn estimation as described in [HON12]. These additional metrics can be easily integrated into both the greedy and DP-based path discovery algorithms.

6.3.3.5 Elevation estimation robustness

As described in Section 6.2.1, it is not always possible to achieve a high accuracy *absolute* elevation estimate. Our path search routines remain robust to errors in elevation estimation in two ways: first, the greedy-based can be operated in *relative* elevation mode, in which each agent is assumed to start with zero elevation error and only deviations from the starting elevation are considered. Additionally, the DP-based search routine can be instructed to search over a range of possible elevation offset values, $\alpha(t)$. In doing so, the minimum score of all paths from all offset values is reported. This inevitably reduces the accuracy of the prediction algorithms, but it allows for some error margin in model (6.2).

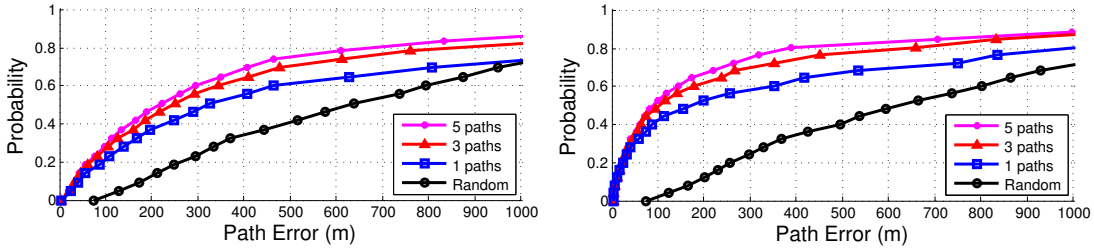
6.4 Evaluation

In order to evaluate the performance of our path prediction algorithms, we collected real driving data and performed extensive simulations over a wide range of geographical landscapes.

6.4.1 Tests on Real Driving Data

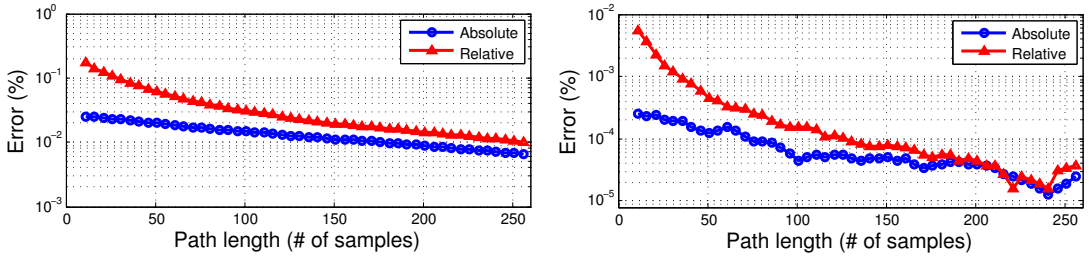
We collected real driving data across 150 km, totaling 4.6 hours of driving time and covering a range of different map topologies. Data was collected using a Nexus 5 with a maximum sampling rate for the barometer (30 Hz) and GPS (1 Hz) for ground truth.

The results of the two path prediction algorithms over all driving data are shown in Figure 6.9. For each, we plot the CDF of prediction root-mean-square errors (RMSE) for a number of ranked paths versus the average error induced by a random walk. More specifically, for each point on the predicted path, we calculate the squared distance to the closest point on the true path. We calculate the same squared error between points on the true path and those on the



(a) Greedy solver (b) DP-based solver

Figure 6.10: Path prediction errors for simulated driving data



(a) Chicago 9.5km x 9.5km (b) Seattle 9.5km x 9.5km

Figure 6.11: Path length v.s. DTW Uniqueness

estimated path, averaging the errors for both and giving us our final RMSE value. The result from '5 paths' represents the best result from the top-ranked 5 paths as estimated by the solver. The random results are fixed to the minimum error of 5 random walks. Figure 6.9(a) shows that the Greedy solver demonstrates a median of around 800 m error, versus the random walk's error of 1600 m on average. More importantly, about 30% of the cases can be solved to within an error of 400 m. These results can be vastly improved by switching to the jointly optimal, DP-based solver, shown in Figure 6.9(b). These results show that, for the driving data collected, more than 50% of the cases can be estimated to within less than 100 m RMSE with around 90% of cases falling below 200 m error.

6.4.2 Simulation

In lieu of collecting driving data across multiple cities for many hours, we conducted a series of simulated test cases. To begin, we download 92 km^2 regions of road data and corresponding elevation data for 26 high-population cities in the U.S. For each city, we conduct a series of random walks of variable lengths and speeds and with barometer noise modeled using an

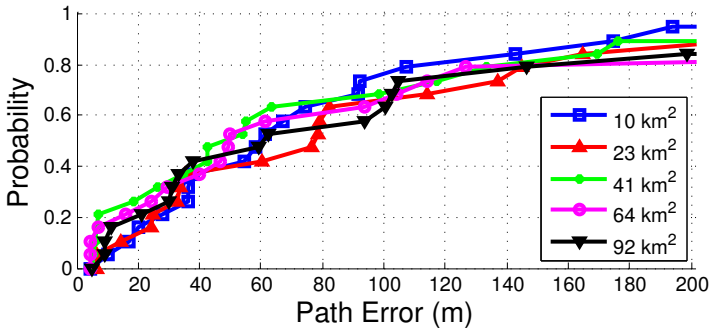


Figure 6.12: Path prediction errors vs. map size.

Ohnstein-Uhlenbeck diffusion process as discussed in Section 6.2.2. These simulated barometer pressures and map/elevation databases were passed to our estimation algorithms in an identical manner to solving the real driving cases. The results of path estimations over these simulated data are summarized in Figure 6.10. Here, the greedy solver shows an improvement over the real driving data, but the DP-based algorithms show a reduced estimation performance. On average over more than 500 simulated test cases, the greedy solver can predict paths to within about 200 m with 50% probability while the DP-based algorithm can predict paths to within 100 m with 50% probability and to within 300 m with around 80% probability.

6.4.3 Analysis of Parameters

The results from real and simulated experiments demonstrated in the previous section indicate that under certain circumstances, driving paths can be quite accurately predicted. In this section, we provide some intuition into factors that affect this prediction accuracy.

6.4.3.1 Path Length

As more barometer data is collected, the probability of distinguishing the correct path from the set of all candidate paths increases. In other words, increased path length typically (though not always) leads to increased path uniqueness. This can be seen in Figure 6.11 for a city with low elevation variation (a particular $9.5\ km \times 9.5\ km$ block in Chicago) to one with high variation (a $9.5\ km \times 9.5\ km$ block in Seattle). For each iteration, we generate two random paths p_a and

p_b with the same length. Path p_n is generated by adding modeled barometer noise over p_a, and a *confusion error* is defined when DTW fails to distinguish the correct, noisy path p_n from the incorrect path p_b. Over multiple iterations of simulation, we observe that an increase in length decreases this confusion error.

6.4.3.2 Map Size

Surprisingly, there does not seem to be a high correlation between map size and path error, as shown in Figure 6.12. This is most likely due to variations in the underlying map's elevation—if absolute elevation estimates can be accurately made, increasing the map size is unlikely to add potential paths whose starting points are of a similar elevation *and* who exhibit similar relative elevation signatures.

6.4.3.3 Geographical Landscape

In addition to variation in the barometer data caused by increased barometer sample sizes, the variation of the elevation in the underlying map also plays a significant role in the ability to accurately predict paths based on pressure. The variations in elevation for the 26 city maps tested in this work are shown sorted in Figure 6.13. Revisiting Figure 6.11, we see that high variation cities like Seattle have a much lower % Error than low variation cities like Chicago. This trend was also observed in general over the 26 cities studied in this work. For example, Figure 6.14 shows the probability of confusing a given random path with any other path in a particular map. As the elevation variation of the underlying map increases (cross-listing again with Figure 6.13), there is an increased chance for path confusion, i.e. an increased probability that any given path may exhibit non-unique elevation signatures.

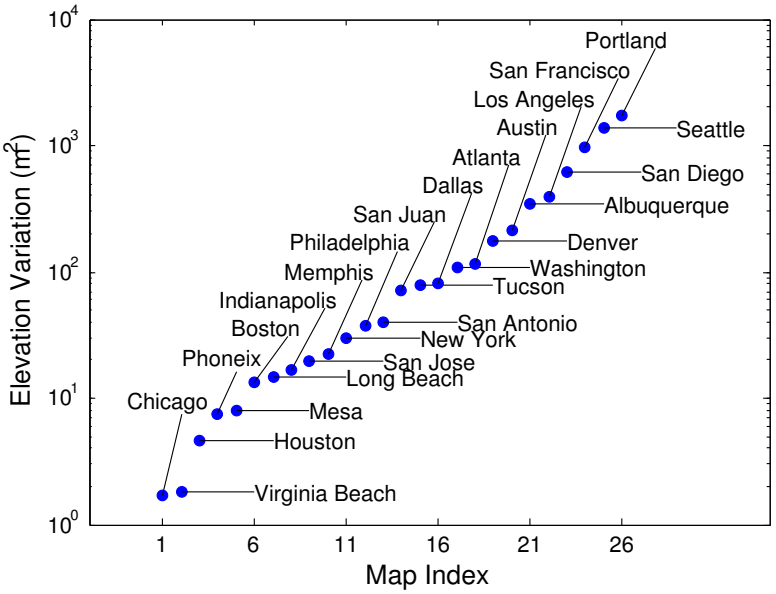


Figure 6.13: Elevation variations for sampled city maps.

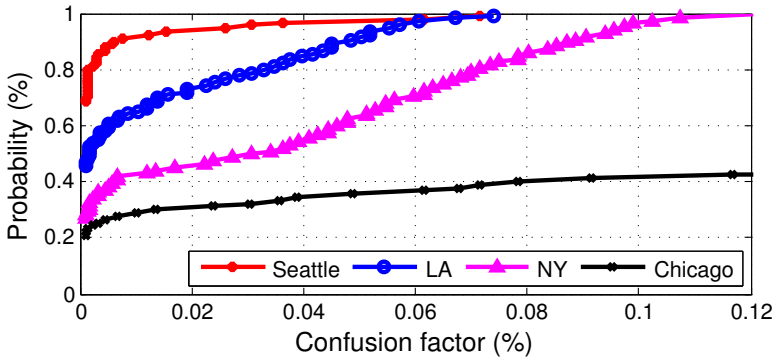


Figure 6.14: CDF of path confusion factors for cities of varying elevation variation.

6.5 Discussion

We have shown through extensive tests in real driving experiments and simulated test-cases that it is often possible to predict a user's driving path with high accuracy from a time-series of barometer data. This prediction, however, is not without its limitations, as discussed below.

6.5.1 Prediction Robustness

As discussed in Section 6.2, the process of converting pressure to elevation depends largely on determining the pressure offset $\alpha(t)$. When this cannot be determined by nearby weather stations, the accuracy will be greatly decreased. This can be counteracted by including additional sensor data such as turn-detection using accelerometers, gyroscopes, and magnetometers. For example, the greedy path estimation algorithm can operate on relative elevation rather than absolute, obviating the need for $\alpha(t)$ entirely. If in addition to using relative elevation estimates we introduce an additional cost in terms of turn similarity, calculated from mobile inertial sensors, the solver is still able to predict 50% of paths to within 500 m RMSE. This may be further improved by considering metrics such as driving speed estimation, driving mobility models, high traffic/highly probable routes, etc.

6.5.2 Privacy Implications

This work demonstrates real driving data in which 90% of tested paths can be predicted to within 200 m RMSE using barometer data alone. Additionally, this accuracy considers only single prediction instances—by combining data across multiple days it is likely that commonly traveled routes can be predicted with a much higher accuracy. With increasingly tight integration of social media applications in mobile devices, the potential privacy risks escalate from associating an *anonymous* user with an estimated driving path to associating a *specific*, personally identified user with a given driving path. When user anonymity, location, and behavior are compromised, the potential for breaches in security and privacy are all-the-more impressive.

Additionally, mitigating privacy leaks through innocuous sensors like barometers may not be as simple as implementing stricter access controls—balancing usability with utility is a non-trivial task, both technically and philosophically. This in essence is the inspiration for efforts like those mentioned in Section 8.1.2.

6.5.3 Future Work

This work demonstrates a privacy leak in the form of barometer pressure backchannels through malicious apps. However, the high correlation between pressure and elevation and the inclusion of these sensors on modern mobile devices raises a number of additional research questions. For example, can barometer pressure be leveraged to improve location-services in real-time to aid in spotty GPS coverage or to further reduce power consumption of location services, as in A-GPS? In addition, can similar methods to those discussed in this work be used to predict pedestrian paths in an unconstrained environment, such as for hikers? Finally, leveraging results describing pressure changes as a function of vertical motion indoors (i.e. elevators, escalators, and stairs) [MKM14], is it possible to infer which building or subset of buildings a user may be walking through based on unique patterns of floor changes, enhancing path detection algorithms and further compromising privacy?

In future work, we plan to explore these questions in an attempt to further evaluate the magnitude of privacy leaks in this regard and to explore the potential for leveraging these results for improved location services.

6.6 Conclusion

We have demonstrated methods by which barometric pressure data collected on mobile phones can be used to infer driving paths with surprisingly high accuracy. Specifically, we described both a greedy graph traversal approach and a dynamic-programming approach to estimating likely driving paths given pressure-based elevation estimates and a map of potential road segments. These methods leverage results from dynamic time warping literature to calculate a rate-independent similarity score between estimated and candidate path elevation signatures. Pressure data collected over a total of 4.6 hours and 150 km demonstrates that these algorithms can predict upwards of 90% of paths with less than 100 m error. Additionally, we illustrated the accuracy of these prediction methods for more than 500 simulated test cases across 26 cities. The results of these simulations show that across all cities more than 70% of paths can be pre-

dicted to within an error of 200 m. We further evaluated the ability to estimate a user's driving path as a function of several variables, including length of barometer data, map size, and the elevation variance of the underlying map.

The results of the methods described in this chapter serve to emphasize the degree to which relative measurements taken on mobile devices can help provide localization estimates with reduced overheads—in this case, with greatly reduced power consumption. In this particular work, this comes at the cost of decreased localization accuracy, but the benefit is several orders of magnitude in power reduction—microwatts rather than milliwatts. In subsequent chapters, we will see that joint optimization need not always decrease accuracy in order to decrease position estimation overheads.

CHAPTER 7

An Open Source Testbed for Indoor Positioning

One challenge in designing and evaluating indoor and mobile positioning systems is that of deploying and providing power to a network of devices, collecting information and measurements from those devices, and obtaining ground truth positions against which to compare any given estimate. In order to obtain ground truth position and orientation estimates, most research institutes use a motion capture such as Vicon [vic] or OptiTrack [opt]. Deploying a network of custom hardware devices, however, is often time-consuming and difficult.

Additionally, deployment overheads and prototyping times are often cited as hurdles to any real-world sensor node deployment [BIS08]. Many indoor positioning systems (and internet of things (IoT) devices as well) benefit from distributed deployments and real-time power measurements, allowing for the developer to evaluate the battery life of their device under test (DUT) quickly and painlessly. To address these problems and to aid in the development of indoor positioning technologies as well as additional projects at UCLA and elsewhere, we have designed an open-source test plaform–the Networked Test Bed, or NTB. This test bed is designed to be a general purpose, distributed deployment and debugging system with the following features:

- **Ethernet power and control**: Each NTB board is powered by a 12W ethernet backbone (using power-over-ethernet, or PoE). The ethernet also provides internet connectivity for an on-board custom TCP/IP server that runs on each device. This server allows for up to 4 clients to connect. Commands include power control, power measurement streaming, ranging measurements, LED control, etc.
- **Independent power measurement on all voltage levels:** Each NTB board can measure power on the 3.3V and 5.0V DUT rails and is capable of streaming these power mea-

surements (voltage, current, and power) to each of four possible clients, using the custom TCP/IP protocol.

- **DUT Control**: Each power rail may be turned off completely or power cycled using the TCP/IP protocol. Additionally, there is a dedicated reset pin for any DUT, and there are a number of other general purpose outputs that can be used to control a DUT in a variety of ways.

- **Expandable daughter-board slots**: Each NTB board has an expansion slot for a "daughter board," providing power and communication with the main processor. This can be used for wireless expansion modules, among other things, and is also made available to the DUT.

- **On-board ARM Cortex M4**: The main processor on each device is an ARM Cortex-M4. This processor manages the on-board ethernet switch and hosts the TCP/IP server. All power measurement, power control, and communication with the daughter-boards is also handled by the main processor.

- **Network Switch**: Each device has an on-board 3-port fast ethernet switch to relay the internet connection to the DUT. Additionally, these network switches are compliant with the latest precision time protocol (PTP) standard, providing support for DUTs that rely on precise time synchronization over the network.

These features are summarized in the block diagram shown in Figure 7.1. Here, GPIO, digital communication, ethernet, power, and reset lines are exposed to the DUT along with a unique 5-bit address programmable via DIP switches on the NTB motherboard. These voltage levels, power measurements, debugging backbone, and ethernet switch management are all managed by the on-board Cortex ARM M4 processor.

Additionally, the NTB motherboard, two revisions of which are shown in Figure 7.4, contains a standard-pitch expansion header for future sensors, communication, etc. This expansion

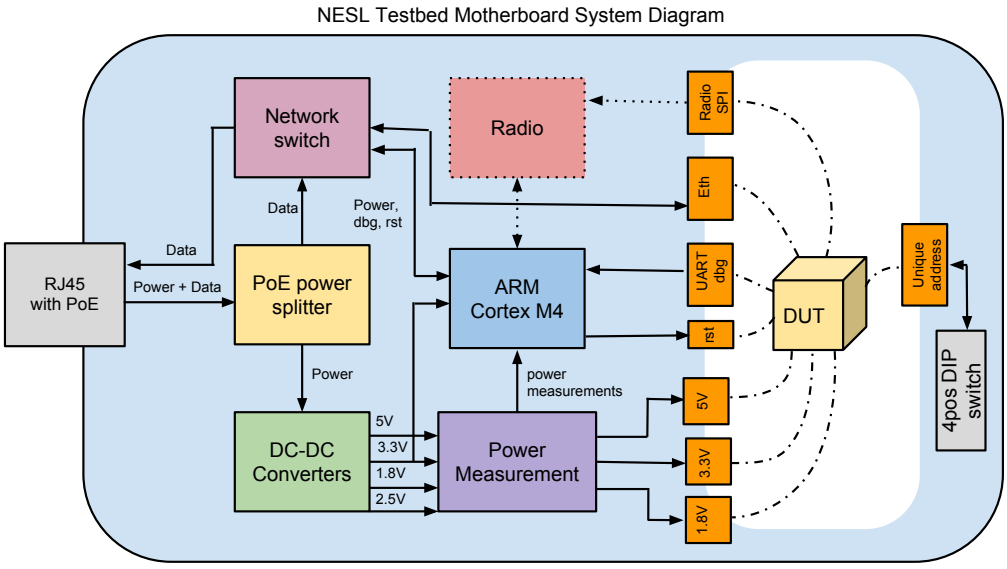


Figure 7.1: Networked Test Bed (NTB) block diagram.

Figure 7.2: NTB Version 1.0 – power over ethernet and DUT hardware support.

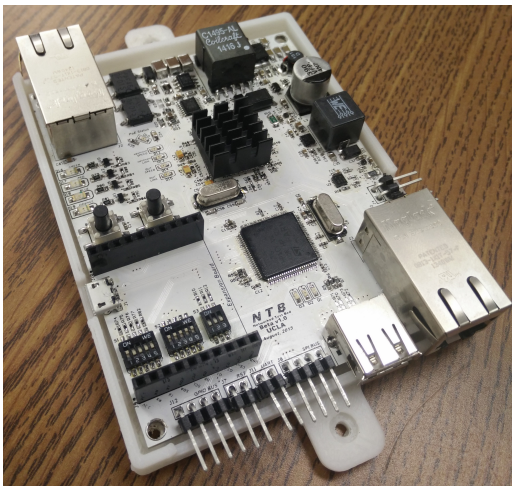


Figure 7.3: NTB Version 2.0 – reduced size and PTP-compliant ethernet switch.

Figure 7.4: UCLA NTB revisions 1.0 and 2.0

Figure 7.5: Decawave DW1000 UWB Expansion board for NTB

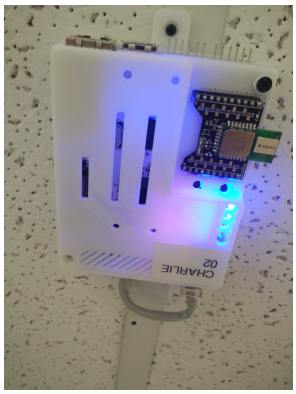


Figure 7.6: A version 2 NTB device mounted to a lab ceiling and powered via ethernet (PoE).

port will initially be used as a port for an ultra-wideband (UWB) radio used for high accuracy time-of-flight distance estimates. This daughter board, shown in Figure 7.5, contains a Decawave DW1000 2.4 GHz radio and is designed to be interoperable with modules designed for small UAV (CrazyFlie 2.0 Quadrotor) platforms. By combining accurate distance estimates from multiple motherboards each equipped with UWB radios, the NTB devices will be capable of accurate indoor positioning, providing decimeter accuracy at a reasonably high frequency update rate. This will enable accurate position estimation alongside power and lifetime analysis for a wide array of indoor localization and positioning-enabling technologies. By making these devices open-source both in hardware and software, we intend to aid additional projects where quick deployment and prototyping stages alongside power measurement are difficult obstacles to overcome. A network of eight of these devices have been affixed to our lab at UCLA (as shown in Figure 8.9) and are used for evaluating the architecture proposed in the following chapters.

7.1 Remote Test Bed Control

In order to facilitate deployment, measurement, and testing of a variety of new devices and infrastructures (particular to but not limited to indoor localization), we have developed a remote control protocol over TCP/IP. This protocol takes a simple 3-byte command of the form

SYSTEM	COMMAND	OPTION

Table 7.1: NTB remote control commands.

Command	Description
	System Control
Cr0	System reset
Ir0	Read unique ID (uid)
	Network Switch
Er0	Reset ethernet switch
El1	Turn on ethernet LED
El0	Turn off ethernet LED
E1<byte>	Set the PTPCFG1_LOW register on the on-board KSZ8463 ethernet switch to <byte>.
E2<byte>	Set the PTPCFG2_LOW register on the on-board KSZ8463 ethernet switch to <byte>.
E3<byte>	Set the PTPCFG2_HIGH register on the on-board KSZ8463 ethernet switch to <byte>.
Ec<byte>	Set the PTPCLK register on the on-board KSZ8463 ethernet switch to <byte>.
	Power Rails
P0<rail>	Turn off voltage rail, where <rail> is 5 (5v) or 3 (3.3v).
P1<rail>	Turn on voltage rail, where <rail> is 5 (5v) or 3 (3.3v).
Ms0	start 100Hz streaming of voltage(mV), current(mA), and power(mW) on the 3v3 and 5v0 lines.
Me0	stop 100Hz streaming of voltage(mV), current(mA), and power(mW) on the 3v3 and 5v0 lines.
	Ranging
Rs0	start streaming of all incoming range measurements.
Re0	end streaming of all incoming range measurements.
Rb<period>	start transmitting beacons with a period of <period> x 100 milliseconds and a dither of 10ms.
Rt<period>	start transmitting two-way-ranging (TWR) from this node. The results will appear on the other nodes when streaming (i.e. after reception of the command Rs0.
	LEDs
L1<led>	turn on led, where led can be the small ARM LEDs: r:red, b:blue, or g:green or the high power, bright LEDs: R:red, G:green, B:blue, O:orange.
L0<led>	turn off led.
Lt<led>	toggle led.

where SYSTEM specifies the NTB subsystem to be controlled (e.g. power, LEDs, ranging) by a given COMMAND, and OPTION is an argument passed to a given COMMAND. In some cases, no option is required and the third byte can be any value. For the sake of completeness, we list several of the more useful commands in Table 7.1. Those commands under the ”Ranging” subsystem are particular to an UWB expansion (daughter) board discussed in the following chapter.

CHAPTER 8

Time and Space

Previous chapters have explored graph realization approaches that incorporate relative range measurements, relative pressure measurements, and relative inertial measurements. In this chapter, we show that even relative *time* can be considered a measurement constraint that can be used to reduce the overhead associated with high accuracy UWB ranging techniques.

Recent advances in ultra-wideband RF communication have enabled accurate packet timestamping, which can be used to precisely synchronize time. Location may be further estimated by timing signal propagation, but this requires additional communication overhead to mitigate the effect of relative clock drift. The additional communication lowers overall channel efficiency and increases energy consumption. This chapter describes a novel approach to simultaneously localizing and time synchronizing networked mobile devices. An Extended Kalman Filter is used to estimate all devices' positions and clock errors, and packet timestamps are measurements that constrain time and overall network geometry. By inspection of the uncertainty in our state estimate, we can adapt the number of messages sent in each communication round to balance accuracy with communication cost. This reduces communication overhead, which decreases channel congestion and power consumption compared to traditional time-of-arrival and time-difference-of-arrival localization techniques. We demonstrate the performance and scalability of our approach using a real network of custom RF devices and mobile quadrotors.

8.1 Introduction

Localization and time synchronization have long been key enabling technologies for wireless sensing, actuation, and communication. With the growing prevalence of wireless devices like those in robotics, home automation, wearables, and mobile smartphones, it is important to coordinate timing among networked devices and to provide contextual information, such as location. Recent advances in ultra-wideband communication (UWB) have enabled decimeter-level localization, but these techniques typically require additional communication to eliminate the effects of timing errors. This introduces a considerable communication and energy overhead for networked devices that are oftentimes resource-constrained.

Time synchronization leverages message exchanges to communicate a global sense of time among multiple nodes in a network—a process where message propagation uncertainties can cause large errors. While localization techniques attempt to mitigate the effect of timing inaccuracies and time synchronization techniques seek to minimize errors due to uncertain device locations, the two are traditionally carried out in a disjoint fashion. This work demonstrates that, by performing localization and time synchronization concurrently, communication overhead can be minimized, increasing the lifetime of battery-powered devices that make up what is increasingly referred to as the *internet of things*.

One challenge in performing time synchronization is that of propagation delay—communication time between two devices is affected by processing delay on the transmitting and receiving devices as well as signal propagation through some medium—e.g. air in wireless systems. For many applications this delay is small enough to be neglected, as is done in several widely used time synchronization protocols such as Reference Broadcast Synchronization [EGE02] and Flooding Time Synchronization Protocol [MKS04]. However, in scenarios where the signal propagation delay is much larger (e.g. acoustic communication, as addressed in [SH06]) or where tighter synchronization is required, signal propagation delay becomes important. Separating the effects of processing delay, propagation delay, and local timing offsets typically requires a bi-directional *handshake* between the devices in question, adding additional timing

constraints that allow errors from propagation delay to be reduced at the cost of increased power consumption and increased channel utilization (e.g., the Timing-sync Protocol for Sensor Networks [GKS03]). In general, time synchronization typically treats this propagation delay as an error term to be minimized.

On the other hand, many state-of-the-art location estimation techniques (and the range estimation methods that underlie them such as ultra-wideband RF time-of-flight) rely on precise measurements of propagation delay in order to estimate the distance over which a given signal has traveled [PAK05]. Whereas time synchronization techniques often treat propagation delay as an error, timing-based ranging techniques commonly treat clock offsets and biases as error terms or "nuisance parameters" [GGS13], attempting to minimize the effects of variations in clock hardware. These variations are particularly detrimental to high fidelity timing-based localization schemes such as time-of-arrival (TOA) and time-difference-of-arrival (TDOA) [STK05, MFA07].

Clearly both time synchronization and localization are closely related, and solving for one necessarily yields information about the other. This motivates the scheme proposed in this chapter, where clock models and device locations are solved for concurrently. This joint optimization provides a principled way of extracting range estimates from time synchronization messages and of using range measurements to refine the time synchronization process based on precise process and measurement models relating timing variations, channel propagation delays, and both motion and clock dynamics. We refer to this technique as **S**imultaneous **L**ocalization **a**nd **T**ime **S**ynchronization, or SLATS. SLATS is implemented as an Extended Kalman Filter (EKF), where the state of each device in a network models time-varying 3D positions, clock offsets, and clock drifts. We evaluate SLATS on custom ultra-wideband wireless devices for networks of both static and mobile nodes, analyzing the convergence time and accuracy of location estimates as well as RF channel congestion and power consumption. Additionally, we demonstrate how SLATS can be used for adaptive messaging schemes, allowing users to specify a desired estimation accuracy (position or time synchronization) at a high level, minimizing communication costs for wireless devices while ensuring that a particular constraint is satisfied.

8.1.1 Contributions

In short, the contributions of this chapter are as follows:

- We propose a method for performing Simultaneous 3D positioning and time synchronization (SLATS) for mobile devices, implemented as a real-time nonlinear state estimator.
- We demonstrate the benefit of SLATS using custom ultra-wideband wireless devices for both static and dynamic networks.
- We further substantiate SLATS using a real-time ultra-wideband quadrotor positioning system compared against motion capture ground truth.
- Finally, we analyze the proposed position and time synchronization estimators in terms of convergence, messaging overhead, and power consumption with respect to state-of-the-art methods.

8.1.2 Related Work

The literature on both mobile device localization and time synchronization is vast. In this section we present only efforts at concurrent synchronization and localization or ranging techniques as they pertain to this work. For more complete treatments of localization and synchronization methods, we refer the reader to [PAK05] and [SY04], respectively.

Recent advances in accurate RF time-stamping, spearheaded by impulse-radio ultra-wideband (IR-UWB) devices [DCF09], has spurred research on low error ranging techniques and their implications on time (and clock) synchronization. One popular application of this has been quadrotor localization (and localization in robotics in general), as demonstrated in [TSW15], [KPD15], and [LHD15] and based on the popular DecaWave DW1000 radio [dw1]. In particular, [LHD15] demonstrates how periodic clock bias measurements can be used to allow for simpler range measurements using one-way TOA/TDOA messages compared to the more expensive two-way protocols used in [TSW15] and [KPD15].

Prior research surrounding jointly optimal localization and time synchronization includes distributed maximum likelihood estimators, theoretical Cramér Rao Lower Bounds (CRLB), batch estimation techniques such as least squares regression, and experimental heuristics. Notable research in the first category includes work by Denis *et al.* in [DPA06] deriving a distributed maximum log likelihood estimator (and corresponding CRLB) that fuses clock offset, bias, and range estimates. Additional work in [ZW10] provides theoretical maximum likelihood and least squares solutions in the case of fixed and time synchronized *anchor* nodes, and similarly [GGS13] presents simulated results for a maximum likelihood estimator for a single target and multiple, synchronized anchor nodes. While these approaches outline theoretical estimators, they make several limiting assumptions of the underlying network (such as synchronization among anchor nodes) and do not evaluate their methods on actual hardware.

Additional methods of integrating range or location estimates with time synchronization include batch (offline) least squares methods [CLV12], master-slave clock synchronization to improve round-trip time estimates [DAZ15], and frequency-tuning crystal oscillators based on estimated frequency errors to improve propagation delay measurements [DTM13]. More application-specific solutions include using unmanned aerial vehicles to relay GPS information to a network of energy constrained nodes who then localize and synchronize themselves to the mobile GPS receiver, saving energy [VGM15], and leveraging clock bias estimation to improve RF time-of-flight estimates for non-UWB transceivers [EMW14].

This work describes a centralized approach to achieving network time synchrony and accurate localization estimates based on concurrent estimation using an Extended Kalman Filter. This can be seen as a centralized version of the work in [DPA06], providing results from real hardware to bolster several ideas presented by by Denis *et al.*. While distributed estimators benefit from improved scalability, the joint estimator presented lends itself more naturally to sensor networks requiring high fidelity, high frequency synchronization and localization, e.g. autonomous robotics and pedestrian indoor tracking. Additionally, tracking the entire network state at a central entity allows for adaptive messaging schemes based on state and covariance values that dynamically constrain the system when estimates begin to diverge or accuracy require-

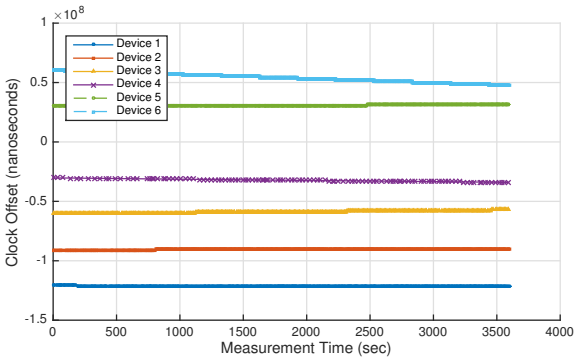


Figure 8.1: Clock offset as a function of time.

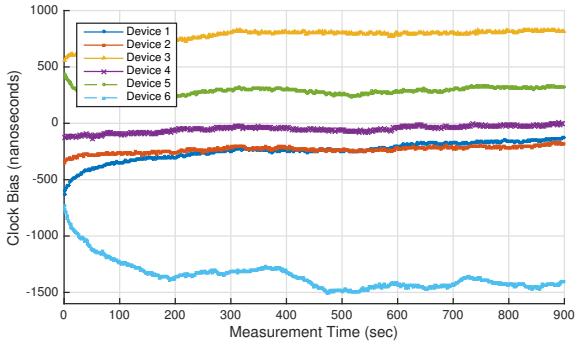


Figure 8.2: Clock bias as a function of time.

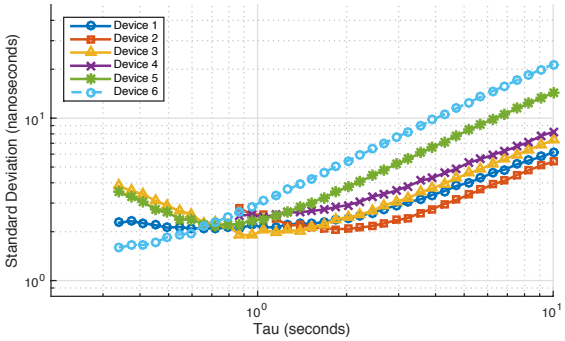


Figure 8.3: Allan deviation of clock bias.

Figure 8.4: Clock errors manifested as time offsets and frequency drifts (biases).

ments change. This work leverages ultra-wideband RF communication in order to make precise timing measurements. While UWB has improved non-line-of-sight performance in comparison to signal strength methods like those based on RSSI, it should be noted that UWB timing accuracy can deteriorate with increased environmental clutter and signal attenuation, as described in [MST15]. While the methods described in this work make no strict assumptions about environmental clutter or line-of-sight, excessive signal attenuation that violates the noise models presented subsequently will undoubtedly affect the state estimates negatively. These models as well as the system dynamics in general are discussed in the following section.

8.2 System Model

We consider a network of N mobile devices (nodes) indexed by $i \in \{0, \ldots, N-1\}$. The state of device $n_i \in N$ at discrete time step k is denoted $\boldsymbol{s}_i(k) = [\boldsymbol{p}_i(k),\ o_i(k),\ b_i(k)]^T$ and is composed

of a three dimensional position vector $\boldsymbol{p}_i(k)$ as well as a timing offset $o_i(k)$ and frequency bias $b_i(k)$. Here and for the remainder of the chapter we use boldface font to denote a vector-valued quantity. We adopt a convention where both $o_i(k)$ and $b_i(k)$ are described with respect to device 0, which we define as the *master time reference*[1]. For relative time synchronization, we can set $b_0(k) := 0$ and $o_0(k) := 0 \ \ \forall k$. However, if projection into some external time reference, such as UTC or TAI, is needed then these values must be synchronized with a secondary source.

8.2.1 Timing State Parameters

Time synchronization and timing-based range estimation rely on accurate hardware time measurement, enabled by high accuracy crystal oscillators, atomic clocks, etc. Each device's clock oscillates close to some nominal frequency f_0. The true signal of every free-running oscillator is perturbed by some frequency bias $b_i(k)$, which varies with time and environmental conditions, such as temperature or supply voltage. This error is often referred to as the *frequency stability* of the oscillator, and quoted in *parts per million*, or $ppm = (f_e/f_0) \cdot 10^6$, where f_e is the worst case upper-bound on $b_i(k)$.

A consequence of frequency instability is that two free-running hardware oscillators will drift with respect to each other unless one of them is periodically *disciplined*. To illustrate this fact we have performed several experiments using the DecaWave DW1000 ultra-wideband radio. This radio is equipped with a temperature-compensated crystal oscillator with $f_0 = 38.4\ MHz$ and a stated frequency stability of ± 2 ppm. Figure 8.4 summarizes the results from an experiment carried out to characterize the stability of 6 different receiver clocks with respect to a transmitter clock. Plot (a) shows the relative time offset $o_i(k)$ of each clock with respect to the master time reference, depending on when each device powered up and on cumulative drift over time. Plot (b) shows that, over short periods, each receiver's clock walks randomly around a frequency bias of no greater than $\pm 2\ ppm$. The accuracy with which this drift can be modeled (and consequently mitigated) is dictated by how much variation exists in $b_i(k)$—in

[1]The selection of device 0 as master is an arbitrary choice that anchors the clock offsets and biases to a fixed point.

other words, how noisy the bias terms in Figure 8.4 (b) are. The Allan Deviation plot in Fig. 8.4 (c) demonstrates the variation present in this bias for a number of estimation periods (tau) and for a given oscillator. The minimum bias variation occurs around a period of 1 second, with variations on the order of 1.0 to 3.0 ppb, or a 1.0 to 3.0 ns hardware clock error over a one second period. Although for many applications this drift is trivial, in the context of RF time-of-flight ranging techniques a deviation of $3\,ns$ is equivalent to a range estimate error of roughly 90 cm. These errors can be largely alleviated through the use of more accurate oscillators such as atomic clocks, but this comes at the cost of prohibitively high power and monetary overheads.

8.2.2 Location State Parameters

Each device state also includes a position estimate $\boldsymbol{p}_i(k) \in \mathbb{R}^3$. The device's true position, $\boldsymbol{p}_i^*(k)$, dictates the propagation delay of any wireless communication between device i and any other node j. More specifically, the propagation delay T_p of RF communication between i and j depends linearly on the distance $\|\boldsymbol{p}_i^*(k) - \boldsymbol{p}_j^*(k)\|_2$. Additionally, any estimate of this range based on T_p is naturally dependent on the non-idealities of device clocks discussed in the previous section. For example, a very basic time of flight ranging technique involves a message sent from i to j, a reply time of T_j measured locally at device j, and then a response sent from j to i. Ideally, the round trip time as perceived by device i would be $T_{\text{round}} = 2T_p + T_j$. Due to frequency bias differences between i and j, however, the measured time would be closer to $T_{\text{round}} = 2T_p + T_j(b_i(k) - b_j(k))$ as perceived by device i's local clock. This will be discussed more completely in Section 8.5.2. Here and for what follows we have adopted the notation that t_x denotes an absolute time measurement while T_x denotes a time *duration*. Finally, networks requiring full 6 degree rigid body estimates can extend $\boldsymbol{p}_i(k)$ to represent the *pose* of device i along with its derivatives.

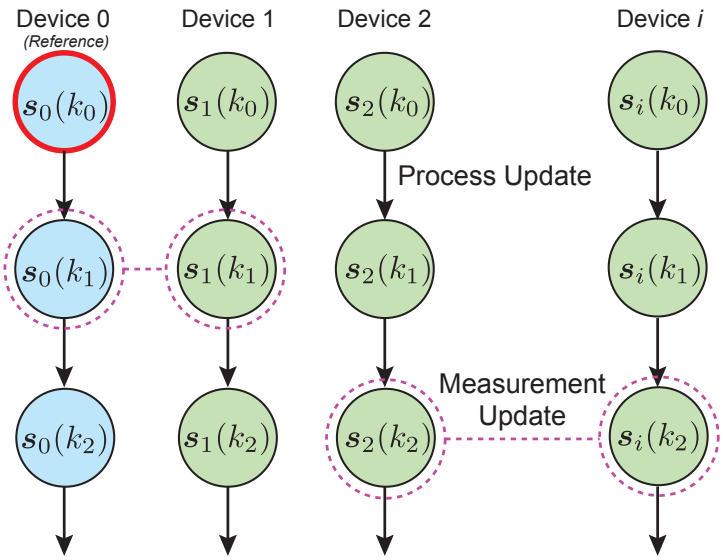


Figure 8.5: A graphical model depicting how the individual device states are evolved over time and periodically corrected by pairwise measurements.

8.2.3 Simultaneous Localization & Time Synchronization

We define the full network state $\boldsymbol{s}(k)$ at master time $t_M(k)$ as the concatenation of all states from the N participating devices at time step k:

$$\boldsymbol{s}(k) = [\boldsymbol{s}_0(k), \;\; \boldsymbol{s}_1(k), \ldots, \boldsymbol{s}_{N-1}(k)]^T \tag{8.1}$$

Given this, we design an Extended Kalman Filter (EKF) in order to evolve $\boldsymbol{s}(k)$ over time, periodically applying corrections when new timing and range measurements are received. Figure 8.5 provides an overview of the filter, where each column represents the evolution of one device's state. Each time index k corresponds to the arrival of a single measurement. Although the measurement at k_1 (row 2) represents messages exchanged between devices 0 and 1, it is necessary that all N device states be updated in order to evolve the network synchronously.

This filter only updates upon arrival of a new measurement (message) exchanged between two or more devices. Section 8.2.3.1 shows how the various measurements are transformed into an observation vector, but for all measurement types the sequence of the filter is as follows:

1. **Receive measurement** - A measurement obtained at filter step k is generated by a message exchange between devices i and j. By convention, communication is always defined so as to terminate at device j.

2. **Calculate** $t_M(k)$ - The server maintains an estimate of master time $t_M(k)$ as well as time offset and frequency bias estimates for each device. This allows us to calculate the time elapsed between measurements in the master time reference.

3. **Process update** - Using $\delta t = t_M(k+1) - t_M(k)$, move the state from $\boldsymbol{s}(k)$ to $\boldsymbol{s}(k+1)$.

4. **Measurement update** - Convert the *measurement* to an *observation* of ranges, frequency biases, and clock offsets, and correct the state.

In the following sections we describe the process and measurement updates required to advance the state estimate based on the message(s) received in step 1.

8.2.3.1 Process update

The filter's process update is responsible for evolving the state estimates for all N devices—clock offsets, frequency biases, and device position or *pose*. In this work we model clock offsets by a first-order affine approximation of the form

$$o_i(k) = o_i(k-1) + b_i(k-1) \cdot \delta t \tag{8.2}$$

where $b_i(k)$ evolves according to a Gaussian random walk. At each iteration, the uncertainty in frequency bias and therefore clock offset for devices without measurements must increase proportional to δt to represent short term frequency instability.

The position or pose estimate of each device must also be advanced according to the prior

state estimate. For a network with *static* device positions, the process update is given as

$$\boldsymbol{s}(k+1) = \begin{bmatrix} \boldsymbol{s}_0(k+1) \\ \vdots \\ \boldsymbol{s}_{N-1}(k+1) \end{bmatrix} = \begin{bmatrix} g_0(\boldsymbol{s}_0(k)) \\ \vdots \\ g_{N-1}(\boldsymbol{s}_{N-1}(k)) \end{bmatrix}$$
$$g_i(\boldsymbol{s}_i(k)) = \begin{bmatrix} \boldsymbol{p}_i(k) \\ o_i(k) + b_i(k)\delta t \\ b_i(k) \end{bmatrix} \tag{8.3}$$

such that each device's position is predicted to be in the same position at each subsequent time step. The covariance matrix corresponding to $\boldsymbol{s}(k)$ is given as

$$P = \begin{bmatrix} P_0 & & \\ & \ddots & \\ & & P_{N-1} \end{bmatrix} \tag{8.4}$$
$$P_i = diag\left\{ \begin{bmatrix} \boldsymbol{\sigma}^2_{\boldsymbol{p}_i} & \sigma^2_{o_i} & \sigma^2_{b_i} \end{bmatrix}^T \right\}$$

where $\boldsymbol{\sigma}^2_{\boldsymbol{p}_i} = [\sigma^2_{x_i} \quad \sigma^2_{y_i} \quad \sigma^2_{z_i}]^T$ when $\boldsymbol{p}_i = [x_i \;\; y_i \;\; z_i]^T$. Note that this simple process model can still be used to capture dynamic node positions if $\boldsymbol{\sigma}^2_{\boldsymbol{p}_i}$ is set adequately high to account for device mobility. For added robustness in networks with highly dynamic nodes, the process update can also evolve $\boldsymbol{p}_i(k)$ using estimates of velocity, acceleration, and orientation which would also be embedded in device state $\boldsymbol{s}_i(k)$. We leave this extension as future work.

8.2.3.2 Measurement update

The measurement update step relates the state variables $\boldsymbol{s}(k)$ to a measurement vector $\boldsymbol{z}(k)$ between two or more devices. For simplicity of notation, these measurements are described with regard to just two devices, i and j, and denoted $\boldsymbol{z}_{ij}(k)$. Each measurement is derived from a message *type* defined by the number of messages sent between devices i and j, as shown in Figure 8.6. Each of these measurements is then calculated directly from the timestamps

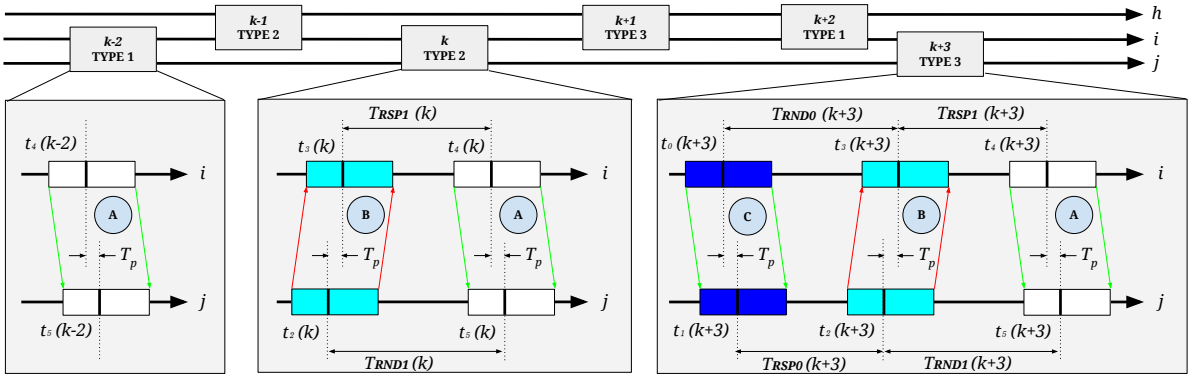


Figure 8.6: The top of this diagram shows six synchronization events between three devices, labeled h, i, and j. Each event is classified as TYPE 1, TYPE 2 or TYPE 3 depending on the number of transmissions sent.

(captured both at the time of transmission and at reception) belonging to these three message types and as described below.

- The **counter difference** d_{ij} is an observation of the difference between the local counters at device i and device j. This measurement includes the effects of propagation delay T_p when sending device i's local counter to device j.

- The **single-sided two-way range** r_{ij} is an observation of the distance between two devices based on a pair of messages between i and j, with error proportional to the response turnaround time T_{RSP1}. This is a noisy range estimate due to frequency bias discrepancies between i and j.

- The **double-sided two-way range** R_{ij} is an observation of the distance between device i and j based on a trio of messages between i and j. The error is proportional to the relative frequency bias between the two devices integrated over the period. This is a more accurate estimate than r_{ij} due to mitigation of frequency bias errors from the additional message.

These exchanges may be of three types: **TYPE 1**, where a single transmission is sent from i to j, yielding two timestamps from a single transmission (TX) and reception (RX); **TYPE 2**, where a TYPE 1 message is followed by a reply message, allowing for round-trip timing

calculations; and **TYPE 3**, where a final transmission completes a handshake trio, allowing for a more precise round-trip timing calculation. These are shown in detail in Figure 8.6, where timestamps $t_0(k)$ through $t_5(k)$ denote the locally measured TX and RX times from the three possible transmissions and $T_{RSPi}(k)$ and $T_{RNDi}(k)$ define response and round-trip durations between the appropriate pair of these timestamps.

It is important to note that a subset of these measurements may be used rather than the full set—in the following sections, we will commonly refer to experiments involving just r_{ij}, R_{ij}, or d_{ij}. Unless otherwise noted, all values are expressed in SI units, with the propagation velocity of radio taken to be the speed of light in a vacuum, denoted c.

The measurement vector, $\boldsymbol{z}_{ij}(k)$, is defined as the following quantity containing all measurement types derived from the three message types above:

$$\boldsymbol{z}_{ij}(k) = \begin{bmatrix} d_{ij}(k) & r_{ij}(k) & R_{ij}(k) \end{bmatrix}^T \tag{8.5}$$

Note that, although we derive the measurements in $\boldsymbol{z}_{ij}(k)$ in a specific manner from the timestamps depicted in Figure 8.6, any measurement of these quantities may be substituted into the SLATS architecture so long as the measurement equations below and the corresponding covariance terms are adjusted accordingly. We define the measurement covariance matrix associated with $\boldsymbol{z}_{ij}(k)$ as the following diagonal matrix

$$R = diag\left\{ \begin{bmatrix} \sigma_d^2 & \sigma_r^2 & \sigma_R^2 \end{bmatrix}^T \right\} \tag{8.6}$$

The measurement vector $\boldsymbol{z}_{ij}(k)$ is constructed as follows:

$$\boldsymbol{z}_{ij}(k) = \begin{bmatrix} t_5(k) - t_4(k) \\ \frac{c}{2}\left(T_{RND1}(k) - T_{RSP1}(k)\right) \\ c\dfrac{T_{RND0}(k)T_{RND1}(k) - T_{RSP0}(k)T_{RSP1}(k)}{T_{RND0}(k) + T_{RND1}(k) + T_{RSP0}(k) + T_{RSP1}(k)} \end{bmatrix} \tag{8.7}$$

For the derivations of r_{ij} and R_{ij}, see Appendix A. Note that the equation describing $R_{ij}(k)$ can be reduced to $R_{ij}(k) = \frac{c}{4}\left(T_{RND0}(k) - T_{RSP1}(k) + T_{RND1}(k) - T_{RSP0}(k)\right)$ if the response times are symmetric.

In order to advance the state of the EKF, these measurements must be compared against predicted values given the current state estimate $\boldsymbol{s}(k)$. This is done by means of the measurement model $h(\boldsymbol{s}(k))$:

$$h(\boldsymbol{s}(k)) = \begin{bmatrix} h_0(\boldsymbol{s}_0(k)) & h_1(\boldsymbol{s}_1(k)) & \cdots & h_{N-1}(\boldsymbol{s}_{N-1}(k)) \end{bmatrix}^T \tag{8.8}$$

Each of these measurements depends on the communication of local clock values between two devices, and thus their derivations naturally depend on both the estimated positions of each device involved as well as their estimated clock parameters. For any given node i in the network, the node-specific measurement model with respect to a second node j is given as (omitting time index k for simplicity):

$$h_i(\boldsymbol{s}_i(k)) = \begin{bmatrix} (o_j - o_i) + \|\boldsymbol{p}_j - \boldsymbol{p}_i\|_2 / c \\ \|\boldsymbol{p}_j - \boldsymbol{p}_i\|_2 + \frac{c}{2}(b_j - b_i)\, T_{RSP1} \\ \|\boldsymbol{p}_j - \boldsymbol{p}_i\|_2 + c \cdot \tilde{R}_{ij} \end{bmatrix} \tag{8.9}$$

$$\tilde{R}_{ij} = \frac{(b_i - b_j)(T_{RND0}T_{RND1} - T_{RSP0}T_{RSP1})}{(1 + b_i - b_j)T_{RND0} + T_{RND1} + T_{RSP0} + (1 + b_i - b_j)T_{RPS1}}$$

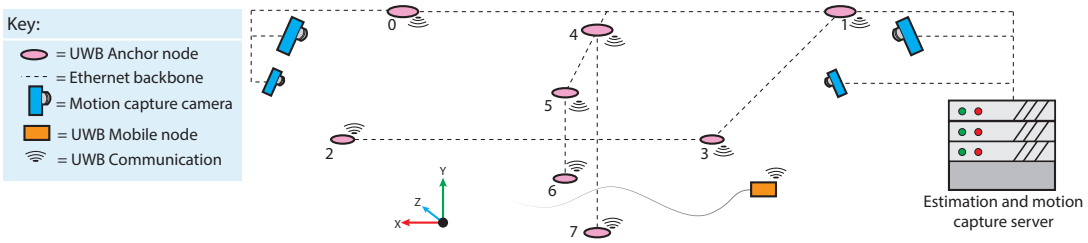


Figure 8.7: Experimental setup overview, including (i) UWB Anchor nodes, (ii) motion capture cameras, (iii) mobile quadrotor UWB nodes, and (iv) an ethernet backbone with centralized server.

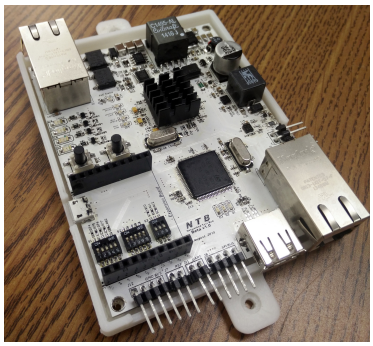


Figure 8.8: Custom anchor node with ARM Cortex M4 processor and UWB expansion.

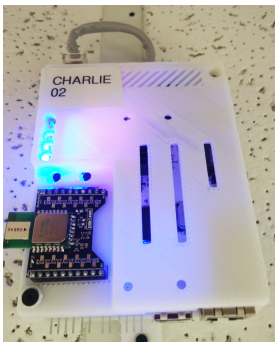


Figure 8.9: Ceiling-mounted anchor with DW1000 UWB radio in 3D-printed enclosure.

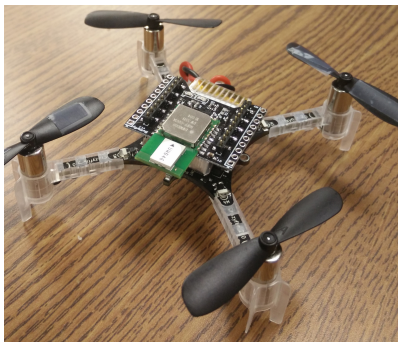


Figure 8.10: CrazyFlie 2.0 quadrotor helicopter with DW1000 UWB expansion.

Note that Equation (8.9) ignores the effects of clock drift *during* signal transmission—a sub-picosecond quantity for distances below 100 m. Interestingly, because TYPE2 and TYPE3 messages are nothing more than multiple TYPE1 messages, range estimates r_{ij} and R_{ij} can be thought of as instantaneous range measurements with a particular covariance, while d_{ij} will constrain range estimates over a larger time span, dictated by σ_d^2. In the following section, we will demonstrate how d_{ij} measurements alone suffice to converge on position, clock offset, and frequency bias estimates for the entire network, provided sufficient connectivity and given constraints on node mobility.

8.3 Experimental Setup

In order to evaluate the performance of the SLATS estimation filter, we developed a custom ultra-wideband RF test bed based on the DecaWave DW1000 IR-UWB radio [dw1] and shown in Figure 8.7. This test bed consists of four main components: (i) fixed *anchor* nodes powered by an ARM Cortex M4 processor with Power over Ethernet (PoE) and an expansion slot for a custom daughter board containing the DW1000, as shown in Figure 8.8; (ii) battery-powered mobile nodes also with ARM Cortex M4 processors based on the CrazyFlie 2.0 helicopter [cra] and equipped with the very same DW1000 radio; (iii) A motion capture system capable of 3D rigid body pose estimation with less than 0.5 mm error; and finally (iv) a centralized server for

aggregation and processing of UWB timing measurements and ground truth position estimates from the motion capture camera.

UWB Anchor Nodes: We deployed 8 UWB anchor nodes in various positions in a 10×9 m lab. Of these 8 anchor nodes, 6 were placed on the ceiling (roughly 2.5 m high) and 2 were placed at waist height (about 1 m) in order to better disambiguate positions in the vertical axis. Each anchor node is connected to an Ethernet backbone both for power and for communication to the central server, and each is capable of sending messages of type 1, 2, and 3, and is fully controllable over a TCP/IP command structure from the central server. These nodes are placed so as to remain mostly free from obstructions, maximizing line-of-sight barring pedestrian interference.

Mobile UWB Nodes: Mobile nodes consist of a CrazyFlie 2.0 quadrotor helicopter platform [cra] with a custom DW1000 expansion board as shown in Figure 8.10. These platforms can be carried around by a pedestrian to enable pedestrian tracking, flown by a user to enable quadrotor helicopter tracking, or flown autonomously by the estimation server if provided with sufficiently accurate position estimates.

Motion Capture Cameras: In order to provide accurate ground truth position measurements, we deployed an 8 camera motion capture system placed along the perimeter of the experimental area. Each camera relays information to the central server, which then computes a rigid body pose estimate. The results presented in this chapter treat this estimate as a "true" position, though we qualify here that all results are accurate to within ±0.5 mm.

Estimation Server: The SLATS state estimation (EKF) is performed on a centralized server. Aggregation of the UWB timing information and ground truth pose measurements is performed using the Robot Operating System (ROS) [QCG09] with custom packages for the UWB hardware used in these experiments.

Coordinate System: Note that, as is typical with camera-based coordinate frames and in order to remain compatible with the motion capture system used, we adopt a right-handed coordinate system where y is the vertical axis and x and z make up the horizontal plane.

8.4 Case Study: Localizing Static Networks

In this section we demonstrate a preliminary case study of the proposed simultaneous localization and time synchronization joint estimator. Specifically, we will examine the case of a network of N static devices—i.e. nodes with fixed positions. This will provide intuition into the various challenges and advantages associated with SLATS, leading into the dynamic networks examined in Section 8.5.

To begin, nodes are placed in 8 distinct locations around the 10×9 m area, roughly in the positions indicated by Figure 8.7. The goal of SLATS is to accurately estimate the positions of all N network devices relative to one another as well as the relative clock offsets and frequency biases. This relative localization, or *graph realization* as it is sometimes called, is a well-researched field. It is well-known that graph realization of nodes in three dimensions is a difficult problem due to local minima, high computational complexity, and restrictions on graph rigidity [AGY04,BY04,JJ05]. SLATS is no exception to the requirements on graph rigidity, and it can also suffer from local minima in certain network topologies. However, SLATS provides a principled approach to estimating a 3D network topology in a manner that can use anywhere from high overhead, high accuracy range measurements (such as R_{ij} measurements) down to low energy beaconing schemes (as in d_{ij} measurements).

8.4.1 Location Estimation

Because the measurement model described in Equation (8.9) relates only pairwise positions, the positions embedded in $\boldsymbol{s}(k)$ at any point represent *relative* location estimates and cannot be compared to any absolute coordinate system. In order to compare the location estimates of each node, we first superimpose the estimated positions onto the true positions of each node by use of a Procrustes transformation [ZW14]. Specifically, the network topology as a whole is rotated and translated (but not scaled) until it most closely matches the true node positions. Once transformed, the error of a given node's position is defined as the ℓ_2 norm of the transformed position minus the true position.

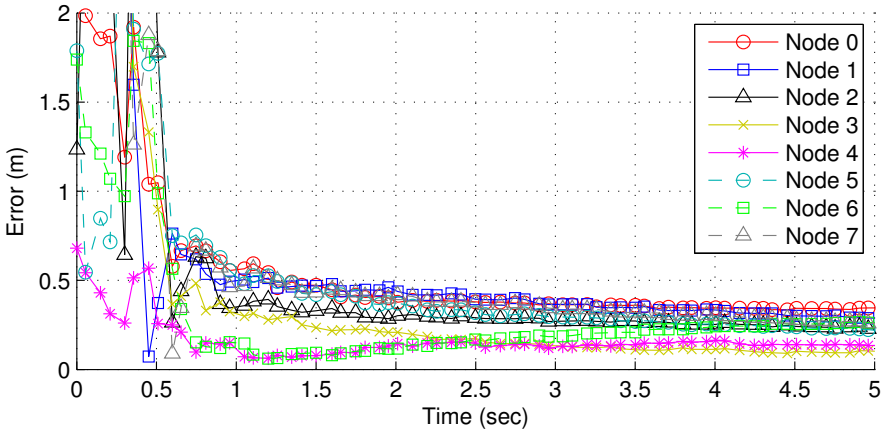


Figure 8.11: SLATS location estimation errors for static nodes using double-sided two-way ranging, R_{ij}.

Table 8.1: Network topology estimation errors for static nodes.

	Node 3D location error by ID (m)								
Solver	0	1	2	3	4	5	6	7	Mean
SLATS: R_{ij}	0.31	0.14	0.16	0.12	0.15	0.13	0.07	0.18	0.158
SLATS: d_{ij}	0.84	0.94	0.63	0.91	0.37	0.17	0.48	0.81	0.644
mdscale: R_{ij}	0.37	0.18	0.33	0.06	0.27	0.16	0.15	0.33	0.231

As a baseline, we begin by demonstrating that double-sided two-way ranging (R_{ij}) can be used to accurately reconstruct (estimate) the topology of the network of 8 anchor nodes depicted in Figure 8.7. We begin with no knowledge of the network topology, naively assigning each node's position to $\boldsymbol{p}_i(k) = [0, 0, 0]^T \forall i \in \{0, \ldots, N-1\}$. Each of the 8 anchor nodes is programmed to initiate a series of double-sided two-way ranging measurement at intervals of 500 ms, or 2 Hz. At each ranging interval, a node will attempt to perform ranging measurements with each other anchor node. Because UWB communication has limited range in comparison to narrow-band communication, nodes at distal corners of the room have sparse connectivity. Results from several 60 second experiments in which all 8 nodes are communicating indicate that the more expensive (higher energy and higher channel utilization) two-way range measurements allow for network topology estimation with high accuracy. Figure 8.11 shows the position error of all 8 nodes as a function of time as the filter evolves, converging after around 1 second. Table 8.1 summarizes the errors of each node after convergence, where node errors range from 7 cm to 31 cm with a mean error of only 15.8 cm. This can be compared against

multidimensional scaling—a commonly used method for solving for a set of coordinates that minimizes some loss function, in this case squared range error based on R_{ij} measurements. The resulting position errors after using 60 seconds of range estimates in a multidimensional scaling solver are shown in Table 8.1. Note that the results achieved from SLATS using only R_{ij} range measurements are comparable to if not better than those using the traditional multidimensional scaling, which has an average error of 23.1 cm.

As noted in Section 8.2, location can also be estimated using SLATS from TYPE 1 timing information alone, due to the interplay between range-induced propagation delay and local timing offsets, $o_i(k)$. If the nodes continue to broadcast their local times at 2 Hz, we can expect to see the effects of frequency bias variation (i.e. the variation depicted in Figure 8.4 (c)) manifest themselves as positioning errors. Specifically, we should expect an error of $[1.5ns, 3.0ns]/2$ or roughly 0.75 to 1.5 ns drift over a 500 ms period. This equates to between 22.5 cm and 45 cm in added positional error when compared to the higher overhead R_{ij} measurements, which are much less prone to timing errors. Indeed, the convergence plot shown in Figure 8.12 and the summarized errors in Table 8.1 indicate that SLATS based on 2 Hz d_{ij} measurements alone gives a position estimation error of between 17 cm and 94 cm, with an average error of 64.4 cm, or roughly 48 cm more than SLATS based on high accuracy R_{ij} range measurements. This lends credence to the ability of SLATS to perform time synchronized (such as TOA or TDOA) localization techniques, with a trade-off between communication overhead (and correspondingly energy consumption) and accuracy.

8.4.1.1 A note on graph rigidity

Note that these anchor nodes compose a fully connected graph, albeit with some packet loss on the more distal nodes. Though a fully connected graph is not required for position estimates, the graph must be *globally rigid* as described in [ST12] in order to yield a unique solution. More specifically, if we create a graph of the N devices where edges represent distance for connected nodes, the graph must be realizable only under translational and rotational transformations—i.e. the relative positions of each node must be uniquely embedded in $\mathbb{R}^3$.

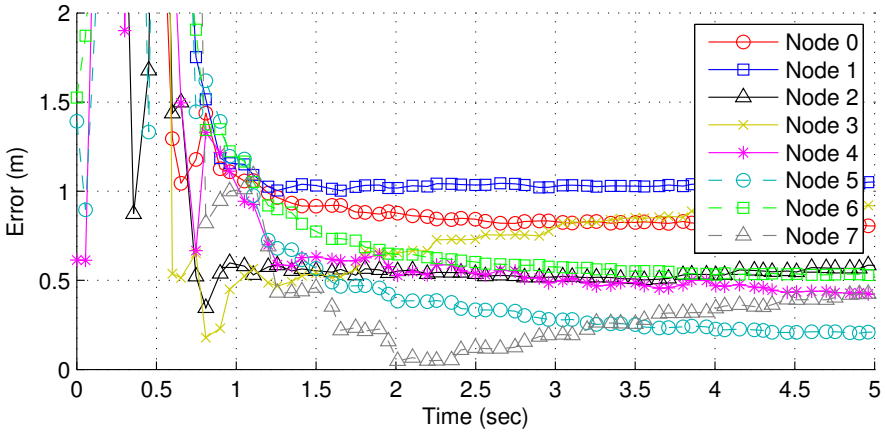


Figure 8.12: SLATS location estimation errors for static nodes using simple timing beacons, d_{ij}.

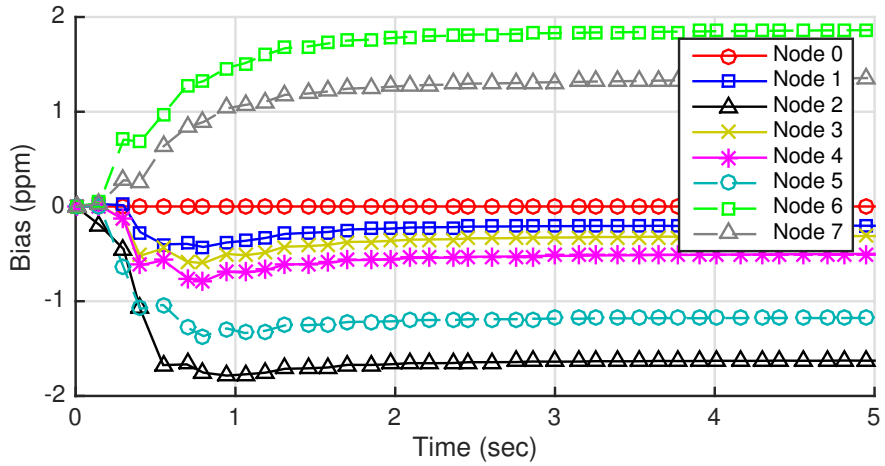


Figure 8.13: Frequency bias convergence for static nodes.

8.4.2 Time Synchronization

The local time synchronization parameters $o_i(k)$ and $b_i(k)$ must necessarily have been accurately estimated to enable sub-meter position estimates from pairwise clock offset measurements, d_{ij}, alone. Due to the precision of d_{ij} measurements, $o_i(k)$ converges within a few time steps. Frequency bias estimated by the evolution of $o_i(k)$, however, takes longer to converge as shown in Figure 8.13. Here, frequency bias estimates take between 1 and 2 seconds to converge, causing the convergence delay in position estimates as seen in Figure 8.12. Note again that the frequency error is bounded by $\pm 2ppm$, reinforcing the results summarized in Table 8.1. Device 0 has both an offset and a bias of 0 by definition, as per Section 8.2

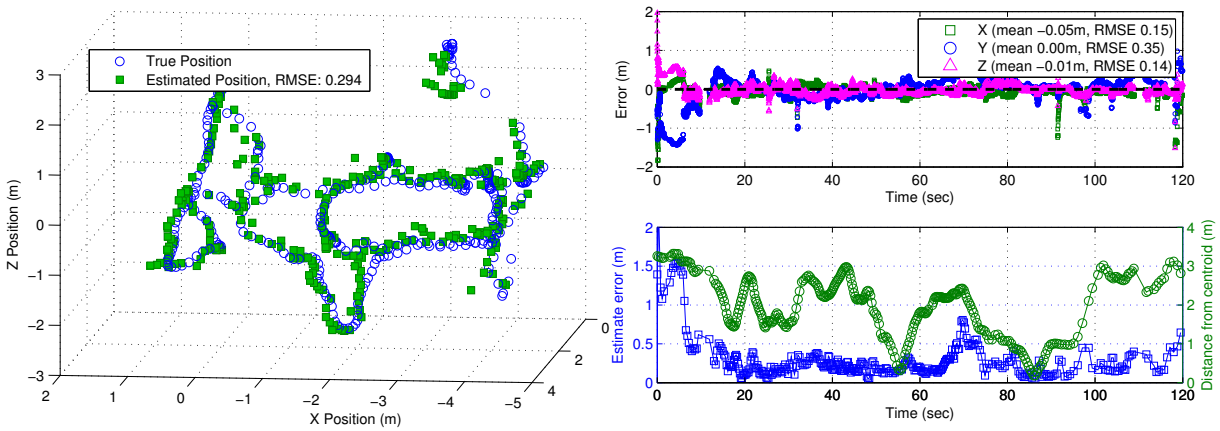


Figure 8.14: Position errors in 3D for a single mobile node. Spatial errors (left) are shown with corresponding per-axis errors by time (top right). Additionally, the error is plotted against the mobile node's distance from the network centroid (bottom right).

Note that, while d_{ij} can be used for both position estimates and time synchronization, R_{ij} is better suited for position estimates than it is for time synchronization, due to its minimal dependency on timing parameters. This, indeed, is the reason R_{ij} is often used in state-of-the-art ranging estimates—it is affected only slightly by errors due to timing asynchrony. As we will show in Section 8.5, one advantage of SLATS is that it allows for simpler ranging schemes like single-sided ranging (r_{ij}) to have comparable performance to R_{ij} with reduced overhead, thanks to the integration of time and range estimates.

8.5 Case Study: Mobile Wireless Networks

Having shown that SLATS may be used to localize and synchronize a network of static, stationary nodes, we now move onto the case of a heterogeneous network of both static and mobile nodes. We begin with the 8 anchor nodes with fixed positions used in Section 8.4, adding one additional node in the form of a CrazyFlie quadrotor as shown in Figure 8.10. Again, we begin by analyzing the results of running SLATS based on R_{ij} measurements. This will provide an *upper bound*[2] on the accuracy that may be achieved with range-based localization methods using the particular ultra-wideband chip-set used in these experiments, representing the state-of-the-

[2]We use this term loosely here, as this is the method used in state-of-the-art UWB ranging.

art in UWB TOF ranging. We will then continue to demonstrate how SLATS may be used to reduce the high overhead and energy consumption associated with these ranging measurements by leveraging the time synchronization provided by SLATS. Finally, we will demonstrate how SLATS can enable low-energy, scalable localization solutions such as TDOA and how SLATS can be used to conditionally and adaptively drive message transmissions in a network based on variable performance requirements.

8.5.1 High accuracy, high overhead positioning

We performed a number of experiments with both pedestrians and quadrotors traveling with variable velocities. Again, each node sends messages at a rate of 2 Hz and SLATS is used to estimate position and time synchronization.

Figure 8.14 shows the result from one particular 2 minute experiment with a single quadrotor whose position is estimated using SLATS based on only R_{ij} measurements. The left plot shows a 3D comparison of the true (as measured by the motion capture cameras) and estimated positions. Here the position can be estimated quite accurately, with an RMSE of just 29.4 cm. The top right plot in Figure 8.14 demonstrates this error per axis as a function of time. While the horizontal axes (x and z) have an RMSE of just 15 and 14 cm, the vertical y axis has an error of 35 cm, due to fewer wireless devices distributed in the vertical plane. Finally, the location of the mobile node dictates to some extent the accuracy with which it can be localized relative to other nodes in the network. More specifically, a device whose location is more central to the network (i.e. closer to the centroid defined as the mean of all node positions) is more likely to be fully constrained in terms of its relative position. This correlation can be loosely seen in the bottom right of Figure 8.14, where several high error peaks (around 5 seconds and 70 seconds) occur during times where the mobile node is far from the network's centroid. Note, however, that estimating a position that is distal from the network's centroid does not *necessarily* yield a poor estimate.

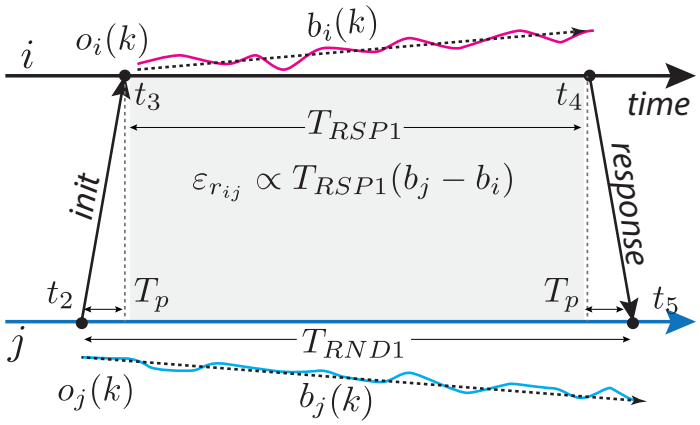


Figure 8.15: Clock drift induced errors in single sided ranging.

8.5.2 Reducing range estimation overhead

While R_{ij} measurements provide precise range (and thus localization) estimates, they introduce a high communication overhead. This section addresses methods of reducing both communication and power overheads by leveraging the SLATS architecture. For each experiment performed, we record the raw timestamps associated with the transmission and reception of each message. This allows us to fairly compare different messaging and estimation schemes on the same dataset.

8.5.2.1 Bias-compensated single-sided ranging

Recall from Section 8.2 that r_{ij} requires fewer message exchanges (TYPE 2) between nodes i and j—33% less to be precise—at the cost of increased range errors, $\varepsilon_{r_{ij}}$, due to timing errors as illustrated in Figure 8.15. Two-way range measurements reduce these timing errors by adding additional measurements, but this timing error can also be reduced by using d_{ij} measurements to synchronize all nodes, after which the error induced by unknown frequency bias terms $b_i(k)$ and $b_j(k)$ can be largely eliminated. This in fact adds no additional overhead, because r_{ij} ranges are constructed from multiple timing measurements, and thus the time difference d_{ij} can be extracted from r_{ij}. Interestingly, because r_{ij} itself depends on $b_i(k)$ and $b_j(k)$ as per Equation (8.9), it can be used to estimate both the positions and frequency biases of each node. Thus, r_{ij} itself becomes sufficiently free of timing errors when passed through the SLATS

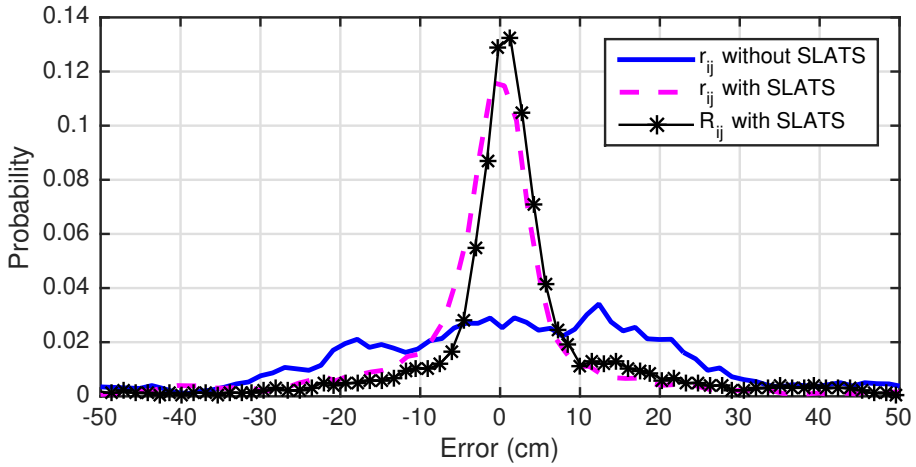


Figure 8.16: Range improvements with bias-compensation.

estimation filter, becoming comparable to double-sided two-way range measurements, R_{ij}. This is shown by the range error distributions in Figure 8.16, based on more than 10,000 pairwise range measurements between 8 devices. Here, the RMSE of r_{ij} measurements drops from 33 cm to 22 cm, matching an RMSE of 22 cm for R_{ij} measurements.

As before, each node communicates at a rate of 2 Hz, but we now restrict communication to TYPE 2 messages. Without using SLATS (i.e. where Equation (8.9) has been changed so that $r_{ij}(k)$ is expected to exactly equal $\|\boldsymbol{p}_i(k) - \boldsymbol{p}_j(k)\|_2$), we observe a 3D positioning error of 65 cm across multiple experiments. On the other hand, SLATS recognizes that the additional error when using $r_{ij}(k)$ measurements is due to the discrepancies in frequency biases, $b_i(k) - b_j(k)$, estimates this difference, and thereby reduces the range estimation errors. This results in a position estimate RMSE per axis of 17, 33, and 18 cm—a 27% error reduction on average and resulting in an estimate that is only 1.3 cm per axis worse on average than the more expensive two-way ranging. The final average 3D error across multiple experiments using the SLATS-enabled r_{ij} measurements is 41 cm.

8.5.2.2 Time of Flight Positioning

As shown in Section 8.4, SLATS can converge on an accurate static network topology (relative position) estimate provided with only simple messages containing the transmission timestamp

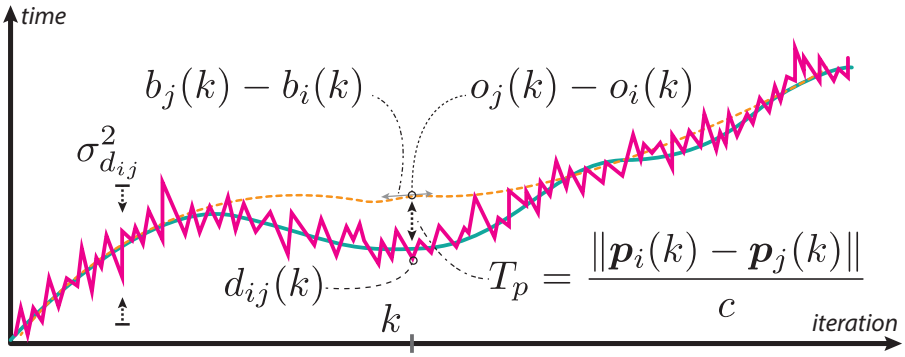


Figure 8.17: Time-of-flight based ranging with SLATS.

and the corresponding reception timestamp, creating the time difference measurement $d_{ij}(k)$ for a message sent from node i to node j. In this section, we'll demonstrate how these simple measurements can also be used to provide a low overhead location estimate for mobile nodes in a network.

Intuitively, the SLATS architecture allows for time-of-flight based positioning estimates enabled by time synchronization and models relating time and space, as in Equation (8.9). Algebraically, extracting range estimates from time-difference measurements is allowed by (i) the polarity inherent in $d_{ij}(k)$ measurements, (ii) accurate timing measurements, and (iii) accurate clock drift models. To explain this, we can derive the singe-sided range measurement from d_{ij} measurements and constants as follows:

$$\begin{aligned} r_{ij}(k) &= \frac{c}{2} \cdot (T_{RND1} - T_{RSP1}) = \frac{c}{2} \cdot (t_5 - t_2 - t_4 + t_3) \\ &= \frac{c}{2} \cdot (d_{ij}(t_5) + d_{ji}(t_3)) \end{aligned} \tag{8.10}$$

This disambiguation between clock offset and propagation delay is shown in Figure 8.17. Single-sided range estimates have increased error due to clock drift at node i over the period T_{RSP1}. Because of this, T_{RSP1} is traditionally kept as small as possible. However, even with a response time as low as a few hundred microseconds, 10s of centimeters error can be accumulated. In the case of periodic $d_{ij}(k)$ broadcasts, however, T_{RSP1} becomes the time between messages and can measure anywhere from milliseconds to seconds apart. In this case, we must rely on accurate bias estimates $b_i(k)$ and $b_j(k)$. Furthermore, in order to constrain pair-

wise distances between any two nodes, we must ensure that the timing measurement and clock offset noise are less than signal propagation uncertainty due to device mobility. Specifically, estimations of the relative clock offset $o_j(k) - o_i(k)$ are subject to measurement noise σ_d^2 and nonlinear clock drift unaccounted for by the relative frequency bias estimate $b_j(k) - b_i(k)$. The latter is given by the Allan Deviation depicted in Figure 8.4 (c), or roughly 3-4 ns/s in the worst case. If the delay between measurements $d_{ji}(t_3)$ and $d_{ij}(t_5)$ in Equation (8.10) is denoted T_d, then the timing error $\varepsilon_{d_{ij}}(t_5)$ inherent in measurement $d_{ij}(t_5)$ is roughly bounded by $\varepsilon_{d_{ij}}(t_5) < 4 \cdot 10^{-9} T_d + \sigma_{d_{ij}}^2 + \sigma_{o_i}^2 \approx 4 \cdot 10^{-9} T_d$, where $\sigma_{d_{ij}}^2 + \sigma_{o_i}^2 \ll 4 \cdot 10^{-9} T_d$ in the case of the UWB transceiver used in this work [dw1]. This places a lower bound on the time synchronization required for a mobile node with a given velocity $\dot{\boldsymbol{p}}_i(k)$—in order for the position displacement due to $\dot{\boldsymbol{p}}_i(k)$ to be less than the error $\varepsilon_{d_{ij}}(k)$, we have $T_d \dot{\boldsymbol{p}}_i(k) < 4c \cdot 10^{-9} T_d$, or $\dot{\boldsymbol{p}}_i(k) < 1.2$ m/s. This is roughly the average walking velocity for a human, which is frequently quoted as 1 to 1.5 m/s [KPN96].

To evaluate this, we performed an experiment with eight static nodes and one mobile quadrotor, each transmitting TYPE 1 messages containing d_{ij} messages at 2 Hz. For this experiment, as with those before, no prior node position or timing information is given—i.e. the node locations are not known *a priori*, even for the static nodes. The results from a single two minute experiment with this setup is shown in Figure 8.18. For this experiment, the vertical y axis is not adequately constrained given the network topology and lack of more accurate range measurements such as r_{ij} and R_{ij} and so it has been omitted from the results, reducing the experiment to one of 2D positioning and time synchronization. In this case, the horizontal x and z axes were correctly estimated to an accuracy of 65 and 56 cm, respectively, with a combined 2D location error of roughly 86 cm. Also shown in Figure 8.18 is the quadrotor velocity as measured by the motion capture system. Note that, barring the initial convergence period, sharp peaks in location estimate errors frequently correspond with velocities in excess of the required 1.2 m/s. Recall that this accuracy is achieved with single transmission TYPE 1 messages sent at a rate of 2 Hz and containing only the transmission timestamp, to be compared against a local timestamp upon reception. SLATS therefore can be used to provide sub-meter position estimates of mo-

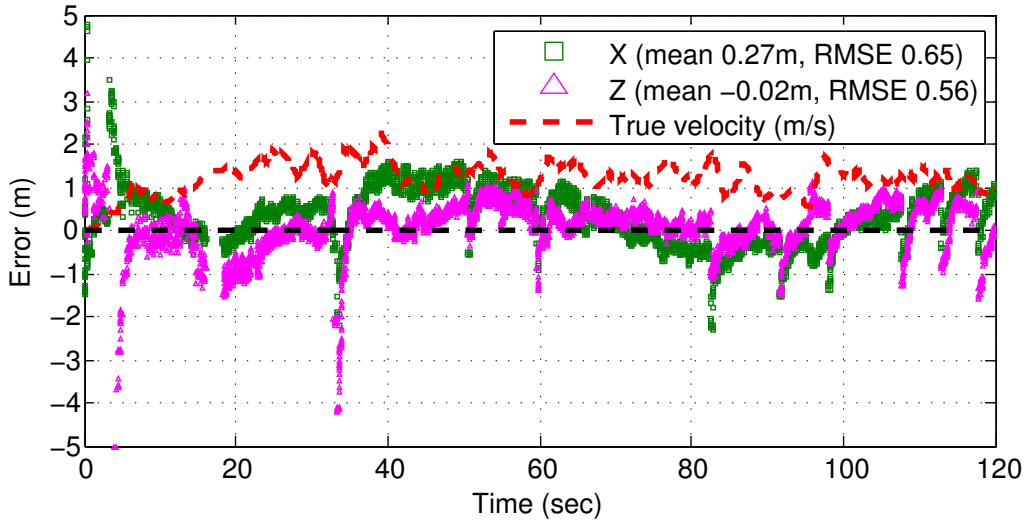


Figure 8.18: Example 2D positioning errors using only pairwise time measurements, $d_{ij}(t)$.

bile wireless nodes, provided sufficient network rigidity and connectivity and where velocities remain less than 1.2 m/s (or the equivalent velocity limit for a given clock source).

Finally, note that increasing communication rates beyond 2 Hz will serve to decrease estimation errors at the cost of increased power consumption and wireless congestion. In order to more easily compare the various methods presented, we limit each to a 2 Hz communication rate.

8.5.2.3 TDOA & Emulating Indoor GPS

Section 8.5.2.2 demonstrated that the time synchronization provided by SLATS allows for time-of-flight based positioning with modest accuracy, but SLATS also allows for time-difference-of-arrival (TDOA) based techniques where the mobile device need not actively synchronize with the remaining network–i.e., where communication with the mobile device is unidirectional. This can take one of two forms: (i) periodic beaconing performed by the mobile node and received at each anchor node, separated in time due to signal propagation delay, T_p, or (ii) staggered beaconing by anchor nodes and received by one or more mobile nodes along with anchor position and timing information. These are discussed in more detail below.

TDOA Beaconing: In this scenario we assume a static network of anchor nodes and one

or more mobile nodes. A given mobile node i periodically broadcasts a message containing its scheduled transmission time, t_i. SLATS is used to provide both localization and time synchronization of all anchor nodes, allowing them to timestamp each received beacon and convert this reception time to the master time by subtracting their current estimated offset. This global time is then used to constrain *pairs* of anchor nodes so as to estimate a position that best describes the disparity in time observed by all anchors. More specifically, an estimate of node i's position $\hat{\boldsymbol{p}}_i(k)$ is obtained by solving the following problem, described in [GG03]:

$$\hat{\boldsymbol{p}}_i(k) = \arg\min_{\boldsymbol{p}_i} \sum_{\ell<m\in M} \left(\frac{\|\boldsymbol{p}_\ell - \boldsymbol{p}_i\|_2}{c} - \frac{\|\boldsymbol{p}_m - \boldsymbol{p}_i\|_2}{c} - T_p^G \right)^2$$
$$T_p^G = (t_\ell^i - o_\ell(k)) - (t_m^i - o_m(k)) \tag{8.11}$$

where ℓ and m are anchors in the set M of all anchors that received the beacon from node i, and t_j^i is the received timestamp at anchor j of the message sent by node i at time t_i. This approach requires minimal overhead for mobile nodes, though it is more advantageous for systems where a central entity is trying to localize a mobile node, rather than a mobile node trying to self-localize. Furthermore, this approach is susceptible to errors in anchor node offset estimates, $o_i(k)$. For several experiments with 2 Hz broadcast rates and using a mobile quadrotor node, this approach yields an RMSE error of 150 cm in the x axis and 123 cm in the z axis, while the vertical y axis had an RMSE of 108 cm.

Indoor GPS Emulation: In a similar manner, we can emulate GPS for indoor environments—here, a mobile node *receives* messages from a subset of anchor nodes. In order to avoid packet collisions, these messages are staggered in time. Because of this staggering, we cannot simply estimate the receiver's instantaneous offset as in Equation (8.11). Instead, we must estimate both the offset and bias terms for a window of received messages. The solution can be achieved

Table 8.2: Comparison of SLATS-based estimation approaches.

Estimation Type	# Msgs	# TX,RX /mob.	Err., (x, y, z)	3D Err.,	$\dot{p}_{max}$	Mob. Power @2Hz
R_{ij}	$3N$	$3N, 3N$	(16, 36, 15) (cm)	30 (cm)	2+ (m/s)	30.5 mW
r_{ij}	$2N$	$2N, 2N$	(17, 33, 18) (cm)	41 (cm)	2+ (m/s)	20.3 mW
d_{ij}	N	N, N	(92, 60, 68) (cm)	119 (cm)	<1.2 (m/s)	9.1 mW
$TDOA_{TX}$	1	$1, 0$	(89, 108, 123) (cm)	190 (cm)	0.5-1 (m/s)	0.2 mW
$TDOA_{RX}$	N	$0, N$	(150, 144, 89) (cm)	212 (cm)	< 0.5 (m/s)	8.9 mW

by solving the following problem:

$$[\hat{o}_i(k) \;\; \hat{b}_i(k) \;\; \hat{\boldsymbol{p}}_i(k)] = \arg \min_{[o_i \;\; b_i \;\; \boldsymbol{p}_i]} \sum_{j \in M} \left(\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2}{c} - T_p^G \right)^2$$

$$T_p^G = (t_j^i - o_j(k)) - (t_i - o_i - \delta_j b_i) \tag{8.12}$$

where M is the set of received beacons at mobile node i for a given time window, and δ_j is the difference in timestamp between a given message and the start of window M. This is the most scalable solution, as there is no limit placed on the number of mobile nodes that can self-localize with the broadcasting anchor nodes. However, this solution also requires a non-linear least squares solver running at each mobile node, and it requires accurate estimates of both offset and bias for each anchor node. Note that this approach is not yet suitable for accurate indoor positioning of highly mobile nodes, as the requirement of batching multiple, staggered anchor messages introduces an inherent delay, and motion within that time window will make it difficult to disambiguate propagation delay and time offset. For a slowly moving node (a human target restricted to less than 0.50 m/s motion), we were able to estimate position using this approach with an RMSE error of 150 cm in the x axis, 89 cm in the z axis, and 144 cm in the vertical y axis. This may be compared to recent work by Vasisht *et al.* in [KJB15], where sub-meter accuracy is achieved using TOF based on WiFi network cards at the cost of comparatively greater power consumption.

Comparison of methods presented: A summary comparison of all mobile tracking methodologies using SLATS and based on multiple experiments tracking both pedestrians and quadrotors is summarized in Table 8.2. This includes number of messages, messaging overhead for mobile nodes, per-axis error, 3D error, suggested mobile velocities, and estimated power con-

Table 8.3: Comparison of SLATS with related work

Estimation Type	Citation	Error	N/m^3	Freq.	Power	$E_m \cdot mW$
SLATS R_{ij}	*this work*	30 cm (3D)	0.029	2 Hz	30.5 mW	9.15
SLATS r_{ij}	*this work*	41 cm (3D)	0.029	2 Hz	20.3 mW	8.32
SLATS d_{ij}	*this work*	119 cm (3D)	0.029	2 Hz	9.1 mW	10.82
PolyPoint	[KPD15]	56 cm (3D)	0.005	20 Hz	300 mW[3]	168
Robotics TDOA	[LHD15]	28 cm (3D)	0.037	100 Hz	443 mW	124
UWB-based GPS	[TSW15]	20 cm (3D)	0.054	10 Hz	152 mW	30.4

sumption by mobile nodes. Power consumption is based on 2 Hz communication power using the DW1000 transceiver and reasonable assumptions on device idle and wakeup times based on the datasheet. This estimate ignores processing power at the mobile node. Table 8.2 shows a very clear trend of decreasing power consumption and decreasing location estimate accuracy. One benefit of SLATS is enabling a developer or system administrator to select *any* of these estimation types, governed by application-specific requirements. In Section 8.6, we will go one step further and show that SLATS enables intelligent and dynamic adaptation across these various positioning schemes in order to meet variable requirements at run time.

We also compare our experimental results with those presented in several notable related works in Table 8.3. Here we omit those purely theoretical results specified in Section 8.1.2– while these provide strong foundations for exploring versions of the methods demonstrated in this work, they lack the empirical results necessary to make a fair comparison. We compare each approach in terms of error, node density, communication frequency, and power consumption. Note that for [TSW15] we have assumed a frequency of 10 Hz, as this is common for GPS units, and this is a UWB-based GPS replacement. Additionally, we introduce the metric of error times power—a good location scheme will have low error achieved at low power, giving a low error-power, while more expensive or poorer localization schemes will have a higher error-power. This is measured in meters times milliwatts.

While [TSW15] achieves the highest accuracy, this comes at the cost of very high deployment density–almost two-fold as much as the deployment used in SLATS with only 10 cm improvement and higher power consumption. Additionally, [KPD15] achieves 56 cm error

[3]This value was calculated in the same manner as those presented in Table 8.2 making identical assumptions but with increased N and a higher frequency.

with a considerably smaller node density, but at the cost of ten-fold increase in communication frequency and a corresponding increase in power compared to the results presented for SLATS. Finally, [LHD15] demonstrates a very impressive localization accuracy for one-way TDOA–28 cm compared to the 212 cm results from SLATS. However, [LHD15] uses 100 Hz communication, a fifty-fold increase over the SLATS results. Because one-way TDOA is quite sensitive to clock drift (and consequently communication frequency), we could reasonably expect SLATS to achieve a similar accuracy with comparable communication frequencies as well. In essence, while related approaches demonstrate a spectrum of power-vs.-accuracy tradeoffs, SLATS allows for one atomic solution that spans the entire spectrum efficiently and adaptively. This is explored more in the following section.

8.6 Conditional Messaging Schemes

One final advantage provided by the SLATS architecture is that of conditional messaging schemes. Inherent in SLATS is a centralized state covariance matrix P maintained and evolved by the underlying EKF. Inspection of the various elements in P yields information about which state variables have uncertain estimates and need additional constraints. This in turn can be used to adjust messaging rates in an adaptive manner, allowing a programmer or user to specify a desired synchronization or localization confidence rather than message type and communication frequency—the effects of which are not always easily predicted. This in turn minimizes communication overheads while guaranteeing that a specified constraint is satisfied.

In order to achieve a desired estimate covariance, we must understand the relationship between the various measurement types and each state variable. For example, if a node i has a poor position estimate $\boldsymbol{p}_i(k)$, the system must know which measurement would best constrain $\boldsymbol{p}_i(k)$—e.g. precise $R_{ij}(k)$ range measurements. Algebraically, for a given state variable $s_\ell(k) \in \boldsymbol{s}(k)$ we may specify a desired covariance term p^*_ℓ such that $p_{i\ell}(k) \in P(k) < p^*_\ell \; \forall \; i$. In order to achieve this covariance, we seek a measurement that will have the greatest effect in reducing the true state covariance after another iteration of the EKF. To determine this, we

can perform a *dry-run* of the SLATS algorithm in order to produce a hypothetical state estimate $\boldsymbol{s}^h(k+1)$ with a corresponding covariance $P^h(k+1)$ with column vector $\boldsymbol{p}_\ell^h(k+1) = [p_{0\ell}^h \ \cdots \ p_{(N-1)\ell}^h]^T$. The problem then reduces to choosing a measurement based on the following:

$$\boldsymbol{z}^* = \arg\max_{\boldsymbol{z}} \boldsymbol{p}_\ell(k) - \boldsymbol{p}_\ell^h(k+1) \tag{8.13}$$

where $p_\ell^h(k+1) \in P^h(k+1)$ is derived from the EKF state equations, omitted here for brevity. In the linear case, we refer the reader to [Say11]. Because $\boldsymbol{p}_\ell(k)$ and $\boldsymbol{p}_\ell^h(k+1)$ are vector-valued quantities, we seek the maximum value of each element-wise subtraction in (8.13). For systems where cross-covariance terms are negligible, $\boldsymbol{p}_\ell$ can be considered as the scalar $p_{\ell\ell}$. In the more general case, however, such as when a node's position must be constrained in 3 axes, we must seek to minimize all elements of the vector quantity $\boldsymbol{p}_\ell$.

As an example, we may want a 95% confidence interval of ± 1 m position error. We can specify $p_i^* = (1/2)^2 = 0.25$ for i being all position state variables—x, y, and z coordinates for all nodes. Intuitively, we expect Equation (8.13) to drive system convergence using accurate $R_{ij}(k)$ ranging measurements whenever the covariance exceeds 0.25 for any position estimate axis. This is shown in Figure 8.19 for an example 2 minute experiment with a mobile quadrotor node, where the error is bounded by ± 1 m (top), and where the quadrotor's positional covariance can be seen oscillating below $p^* = 0.25$ due to R_{ij} range measurements sent when $p_{xyz} > p^*$. At 20 seconds, the quadrotor has poor network connectivity, and p_{xyz} temporarily exceeds the desired covariance.

8.7 Discussion

SLATS is far from the first work to suggest the marriage of time and localization estimates–a cursory read of Section 8.1.2 or any of the vast literature on GPS will say otherwise. Rather, what SLATS introduces is a method by which time synchronization may be fused with timing-based range estimates in order to provide accurate position estimates for commodity and con-

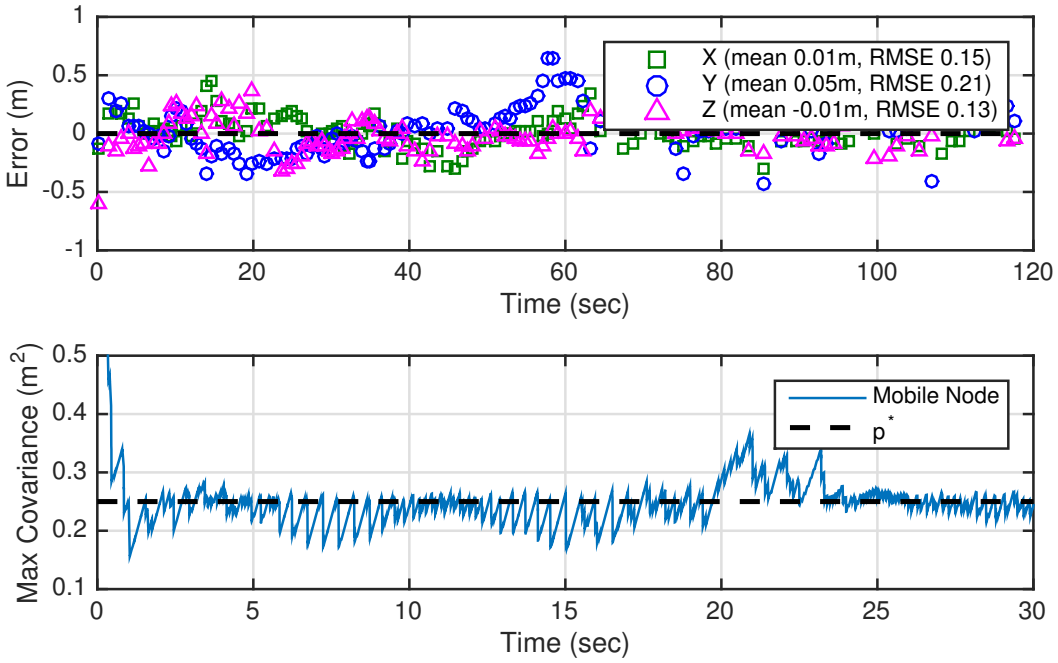


Figure 8.19: Conditional range messaging with $p^*_{xyz} = 0.25$.

sumer electronics at a decreased communication and power overhead with respect to the state-of-the-art. Moreover, SLATS provides an elegant manner for specifying requirements in terms of localization or time synchronization accuracy–inherently tuning a "knob" that trades power for accuracy.

The SLATS architecture proposed in this work is not without several limitations and practical considerations, however. We discuss some of the key considerations in adopting SLATS below.

8.7.1 Centralized and Decentralized Approaches

The SLATS architecture, as presented in this work, requires a centralized estimation entity. This implies that the EKF algorithm must run on a single computational platform. For robotics localization, high fidelity pedestrian tracking, or small-scale (e.g. smart home) deployments, this is often a practical assumption. For large, multi-hop networks, however, a distributed approach would be more favorable. Future work will explore adapting the SLATS architecture to a scheme that is more amenable to large distributed networks

8.7.2 Antenna Orientation

UWB antenna orientation influences message timestamps and consequently range measurement accuracy. We ignore this effect in this work, but the effect can introduce errors in the low decimeters. This is difficult to account for without knowledge of device *poses* for the entire network and without accurate models of 3D antenna orientation vs. signal attenuation. While prior work has focused on characterizing these models (e.g. [TSW15]), it may also be feasible to incorporate antenna attenuation as a state variable. Future work will explore fusing inertial (particularly gyroscopic) measurements with ranging measurements to estimate antenna attenuation at run time.

8.7.3 Device Mobility and Timing Errors

The TOA- and TDOA-based SLATS solutions presented in this work are not well suited for high mobility networks with the hardware and frequencies used in these experiments, as summarized in Table 8.2. These methods will become increasingly viable as UWB hardware timestamping continues to improve, and particularly as crystal stability (ppm) continues to improve, requiring less frequent synchronization messages and allowing for tighter synchronization in general.

8.8 Conclusion

In summary, this chapter has described and evaluated SLATS–an architecture for simultaneous time synchronization and positioning of mobile networks. Our results from numerous experiments using real, custom ultra-wideband wireless anchor nodes and mobile quadrotor nodes indicate that our architecture enables a number of localization strategies—from energy and communication demanding position estimation techniques with 30 cm error, to more scalable, time compensated single-sided range estimation with similar errors, and finally to highly scalable TOA and TDOA approaches with meter-level accuracy. Most notably, SLATS allows for (i) mobile network graph realization, (ii) time synchronization with signal propagation compen-

sation, and (iii) an emulation of GPS solutions in the indoor environment based on TDOA. This research is made possible by state-of-the-art advances in commercial ultra-wideband radios, and continued improvements to these devices will further underscore the importance of treating temporal and spatial variables in a joint fashion.

CHAPTER 9

Collaborative Localization and the Internet of Things

In Chapters 7 and 8 we demonstrated that advances in time measurement in (ultra-wideband) packet radios have enabled accurate position estimation for commodity embedded devices. Because the system described in Chapter 7 is far more practical and affordable than high accuracy, high fidelity motion capture technologies such as Vicon [vic], Optitrack [opt], we are motivated to explore use-cases of this technology in more feasible sensor deployments—specifically, in scenarios where one or more users has placed multiple smart and connected devices in an indoor environment, such as a smart (automated) home or office. As connected smart devices—those commonly referred to as devices making up the *Internet of Things*, or IoT—continue to grow in number, enabling user interaction with these devices in an intuitive and scalable manner will be a considerable challenge.

Consider a smart home with a family of four and several connected devices—controllable lighting, locks, and the like. Current smart devices typically require an application to be installed on a mobile device in order to connect to and communicate with connected devices. Additionally, users typically are burdened with providing semantic labels for each of their devices–"living room light 2," or "north east door," for example. While such labels provide intuitive control over a small number of devices, burdening a user with selecting a device label amongst tens or hundreds of devices will prove cumbersome. This motivates the development of a new technology to allow *accurate* and *scalable* control and communication with smart IoT devices. Towards this, we focus on arguably the most intuitive and natural form of human-device interaction—the "gesture." We discuss recent efforts in this vein in the following section.

9.1 Related Work

Considerable research in the past decade has considered gesture recognition in the smart home setting. These efforts can largely be divided into three groups: (i) wireless-sensing based gesture recognition using electromagnetic fluctuations such as doppler shifts in WiFi signals [AKK14, AK13] and RFID signals [BBB14]; (ii) infrastructure-based gesture recognition such as those employed by Microsoft Kinect [kin] and other depth sensors, and (iii) inertial measurement based gesture recognitions like those detected by smart watches and other wearables [WRD16].

Wireless-sensing based gesture recognition such as the numerous efforts from MIT's CSAIL group (including [AK13]) provide coarse gesture recognition and pedestrian localization in an area covered by WiFi MIMO technology. These efforts have demonstrated modest gesture recognition accuracy, but these techniques do not apply easily to communication between multiple users and multiple devices. Specifically, these techniques are fundamentally limited in their spatial resolution—that is, the ability to detect where and in what direction a gesture is made relative to other distributed devices. Techniques based on depth sensors and cameras, although potentially quite accurate, are typically limited to line-of-sight. Finally, while inertial based gesture recognition on wearable devices can be very accurate and can perform "always-on" sensing, due to continuous coverage of pedestrian motion, there is no inherent way of determining the "target" of a given gesture–that is, if a user would like to turn on a specific light, how should she specify to which light she is referring? In fact, without augmenting the above techniques with a position estimation technique coupled with the predetermined locations of all smart devices, none of the above techniques can be used to directly control a *specific* device based on human gestures, barring techniques that assign a special gesture to each unique device.

We propose a technique that allows specific device control, requiring no device configuration or position estimation. We base this technique on ultra-wideband ranging using the hardware and analysis methods developed over the previous chapters. For this chapter, we consider a user wearing a smartwatch equipped with an inertial measurement unit (IMU) and a UWB radio, arguing for the adoption of UWB packet radios for wearable devices in the near future. The

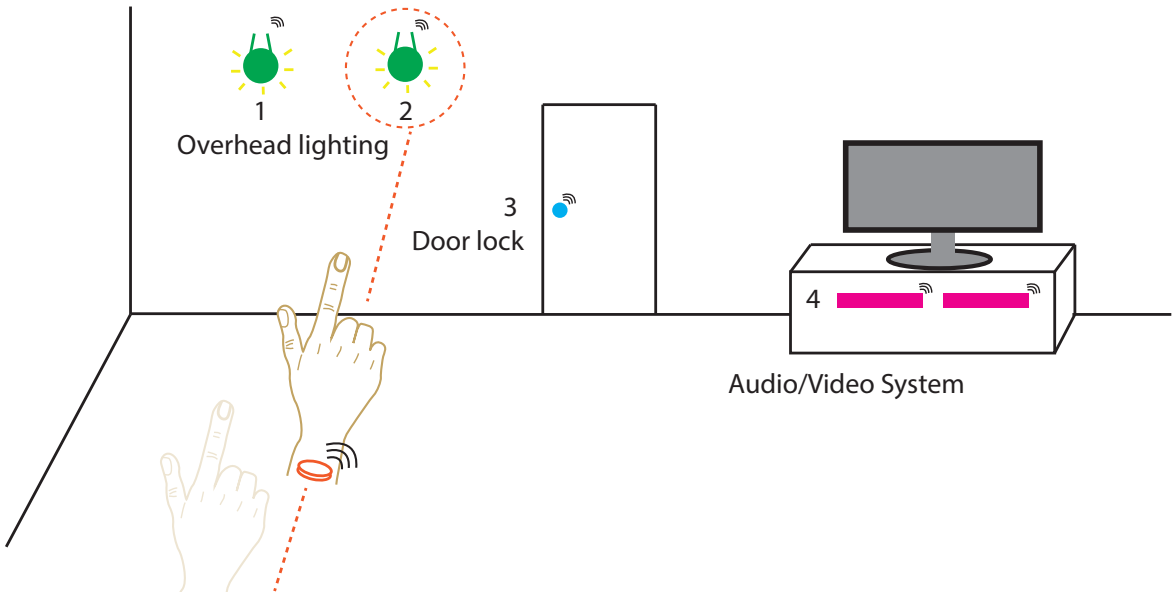


Figure 9.1: Gesture based IoT device selection using wearable devices

description of this IoT device selection technique is the the subject of the following sections.

9.2 IoT Device Selection

Consider a user with a wearable device such as a smartwatch in an indoor environment equipped with a number of smart (and connected) devices. We will assume that the smartwatch and all IoT devices are equipped with a UWB radio—perhaps a strict assumption, but given the success and growing adoption of UWB packet radios, we anticipate that they will permeate the IoT scene in the next few years. An illustration of this set up is given in Figure 9.1 for a few example IoT devies.

We consider a network of N IoT devices and, for the sake of simplicity, a single user u whose wearable device is mobile with (initially) unknown position. We denote the position of a node $n_i, i \in \{0, \ldots, N-1\}$ by $\boldsymbol{p}_i = [x,\ y,\ z]^T$. The location of the mobile user device is correspondingly denoted $\boldsymbol{p}_u$. In order to "select" an IoT device n_{i^*} to control, a user must "point" towards that device. Here i^* is used to denote the desired IoT device the user wishes to control or communicate with. The goal of this pointing gesture is to create a motion such that the distance between n_{i*} and n_u decreases by a magnitude greater than the distance between n_u

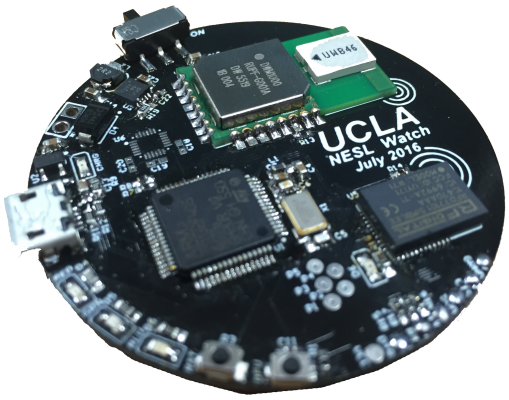


Figure 9.2: NTB "watch" companion. This is an UWB-capable, battery-powered mobile tag for users to carry or wear, allowing for range estimates in real time. The tag is also equipped with Bluetooth LE, sending range and anchor information to supported mobile devices such as smart phones.

and n_i for $i \neq i^*$. If the IoT devices comprise a fully connected graph, communication between all nodes might allow us to detect i^* by solving the following idealistic problem:

$$i^* = \arg\max_i \|\boldsymbol{p}_i - \boldsymbol{p}_u(t_s)\|_2 - \|\boldsymbol{p}_i - \boldsymbol{p}_u(t_f)\|_2 \tag{9.1}$$

where t_s is the time at which the user begins to point, and t_f is the time the user stops pointing (with arm fully extended). Problem 9.1 relies only on pairwise range estimates between a user's (wearable) device and various IoT devices. Unfortunately, range estimation errors even in double-sided two-way ranging techniques disallow a simple selection algorithm like that illustrated above due to a number of factors, namely range noise, pointing "length" and spatial "diversity," as discussed below.

9.2.1 Spatial Resolution & Gesture Length

When a user points (or makes any gesture towards) an IoT device, the length of that gesture inherently defines a signal to noise ratio, or SNR. We denote this length ℓ, and we assume range errors to be non-zero mean Gaussian distributed by $r \sim \mathcal{N}(\mu_r, \sigma_r^2)$. This error is not

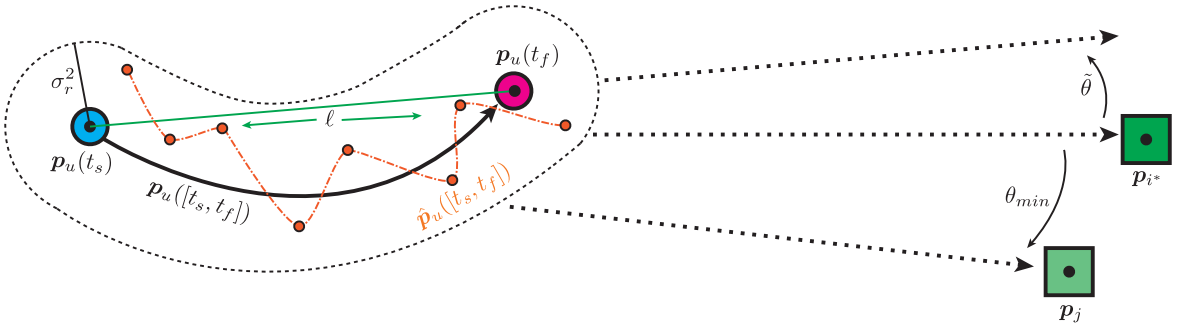


Figure 9.3: Ranging errors and angular (spatial) resolution in gesture-based IoT device selection.

necessarily i.i.d, but for the sake of simplicity we will assume that range error is independent in time. Additionally, when a user points "towards" an IoT device in order to select, control, or communicate with it, the user may point with some angular error $\tilde{\theta}$. Additionally, given the user's true position $\boldsymbol{p}_u$, there exists some minimum angular distance θ_{min} between the desired IoT device n_{i^*} and any other device n_j. In the extreme, this angle could be $\theta_{min} = 0$ radians if a second, unwanted device is co-linear with the user and the preferred device n_{i^*}. In such a case, it is impossible to accurately detect which device a user wants to communicate with using gestures alone. These various errors and parameters are illustrated in Figure 9.3. In the case of high spatial diversity—i.e. when θ_{min} is quite large—and when ℓ is large with respect to σ_r^2, it is reasonably easy to distinguish between the device a user is pointing at, n_{i^*}, and any other device n_j. An illustration of the range measurements between user and all IoT devices—i.e. estimates of the quantity $\|\boldsymbol{p}_u - \boldsymbol{p}_{i=\{0,1,\ldots,N-1\}}\|_2^2$—during the duration of a single "pointing event" with high spatial diversity is shown in Figure 9.4. In this illustration we distinguish between "True Traces" in red—those corresponding to the distance estimates between user and n_{i^*}—and "False Traces" in blue—those corresponding to range estimates between the user and any other device. Clearly the two traces are easily separable using the difference in range estimates $\Delta R = \|\boldsymbol{p}_i - \boldsymbol{p}_u(t_f)\|_2 - \|\boldsymbol{p}_i - \boldsymbol{p}_u(t_s)\|_2$. Inspection of ΔR shows that an average point length ℓ ranges from about 30 cm to about 80 cm, depending on the length of the arm and the gesture itself. From earlier chapters, we demonstrated range error magnitudes in the range of $\sigma_r^2 = 10 - 30$ cm using the specified UWB radios.

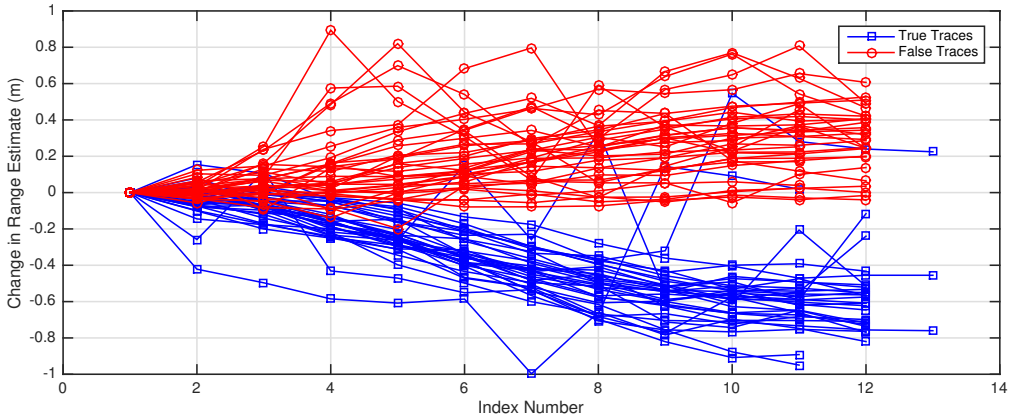


Figure 9.4: Example range traces during one event of a user "pointing" at a device equipped with UWB ranging technologies. This example is in an environment with *high* spatial diversity.

In the case of low spatial diversity in the face of non-ideal range measurements, the simple metric ΔR no longer suffices as a reliable metric for disambiguating desired IoT devices and other IoT devices. As an example, Figure 9.5 shows estimated range differences for a low spatial diversity environment—in fact, the same environment depicted in Figure 8.7. In this case, the true range measurement traces—those measuring the distance to the desired IoT device—follow a reasonably predictable pattern. Those measuring distance to all other IoT devices, however, have high variance, making it more difficult to distinguish between "true" and "false" range traces.

In these cases, it is perhaps easier to look at the distance metric ΔR itself. This is depicted in Figure 9.6, where the overlap between True and False range differences is readily obvious in the range of -0.5 to -0.8 m. Clearly, in order to provide high accuracy device selection we need to explore additional metrics. In the following section, we will explore a more principled approach to the device selection problem, keeping device power and computational overheads in mind.

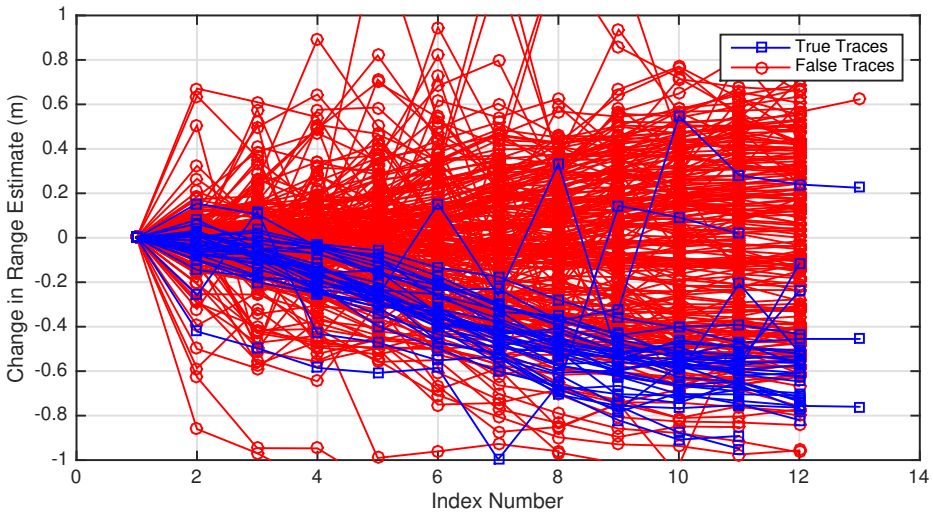


Figure 9.5: Example range traces during one event of a user "pointing" at a device equipped with UWB ranging technologies. This example is in an environment with *low* spatial diversity.

9.3 Device Selection by Pattern Matching

We've demonstrated thus far that a single metric of estimated distance changes, ΔR, is insufficient in reliably detecting the device a user is pointing towards. This motivates the use of additional metrics in distinguishing desired and undesired devices. Several of these are discussed below.

Linear Fit:

In the ideal case, a user will point directly towards the device that she wants to select, following a nearly linear path. In figure 9.3 we have shown this path to be curved to illustrate that in general this may not be true, but in most cases the pointing gesture can be thought of as a path from $\boldsymbol{p}_u(t_s)$ to $\boldsymbol{p}_u(t_f)$. Any divergence from this fit could indicate that the range estimate trace could belong to an undesired device. More specifically, if we estimate a linear fit $\hat{r}_{ui}(k) = \|\boldsymbol{p}_u - \boldsymbol{p}_i\|_2 + v_k$ describing the range estimate between n_u and n_i for a given device i and noise v, we describe the *badness* of fit as the mean squared residual—$msr = \frac{1}{k}\sum_k v_k^2$. Intuitively, a higher msr should indicate a poorer linear fit and consequently a less likely trace "candidate."

Path Loss:

The accurate timing provided by UWB radios is enabled by measuring the energy in the radio's

accumulator corresponding to the communication along the first path. One consequence of this accurate timing is that the same energy can be used to estimate the power of the communication along the "first path" of communication—typically the line-of-sight path. Intuitively, a user is likely to point to a device that is within line-of-sight, so the correct (desired) device is likely to have low first path loss, which we denote $fploss$ and measure in dBm.

Beginning Range:

Similarly to $fploss$, it is likely that, if a user is pointing towards to co-linear devices, the desired device is the closer of the two. Because of this, we give some weight to selecting the closest device.

Angular Divergence:

As a user points from $\boldsymbol{p}_u(t_s)$ to $\boldsymbol{p}_u(t_f)$, he is attempting to point as closely *towards* a device as possible. In an ideal case, the line between start and finish is perfectly co-linear with the coordinates of the IoT device itself. In practice, however, there is some angular divergence $\tilde{\theta}$ as shown in Figure 9.3. In order to calculate this angle, however, we must know the device position $\boldsymbol{p}_i$ as well as the start and stop position of the user's device. This requires a collaborative position estimate among multiple IoT devices, which assumes a certain device density and additionally consumes more power due to added communication costs. When possible, however, $\tilde{\theta}$ can be used to increase the accuracy of gesture-based IoT device selection.

9.3.1 Feature Vectors

We combine the previous "features" into an aggregate feature vector, defined as

$$\boldsymbol{f} = [msr, fploss, \hat{r}(t_s), \tilde{\theta}]^T \tag{9.2}$$

Here we have included the expensive feature $\tilde{\theta}$, but we will also demonstrate results that omit this feature in order to provide more realistic results for sparse deployments and restricted energy reserves. For each pointing event, there will be N features constructed and correspondingly N feature vectors, barring communication errors. Of these, 1 feature vector will belong

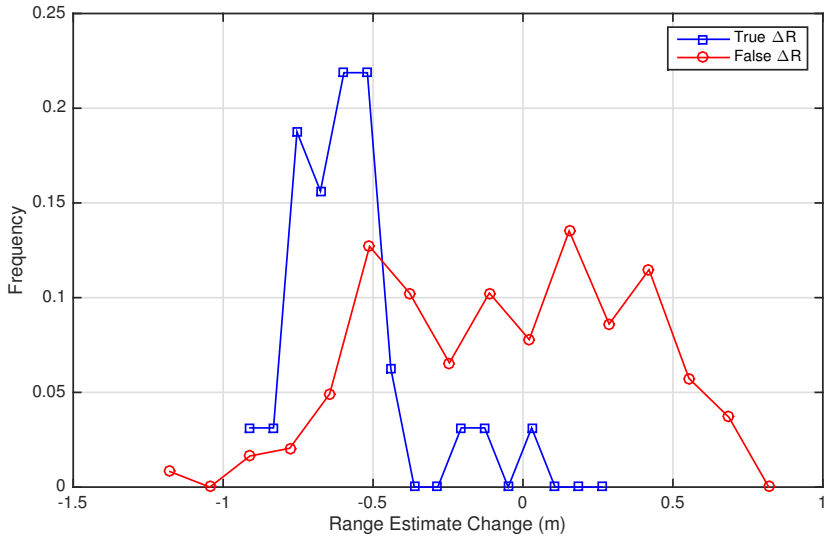


Figure 9.6: Range difference, ΔR, for true and false (n_{i^*} and $n_{i \neq i^*}$) as a probability density function.

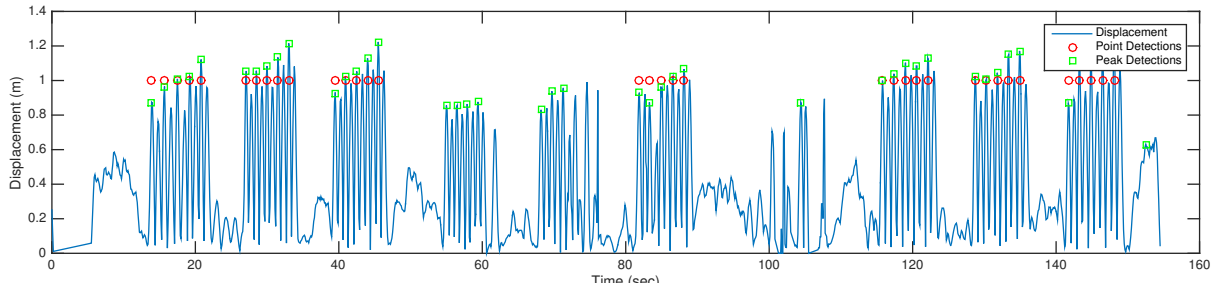


Figure 9.7: Motion capture ground truth calculation for IoT gesture-based device selection. Results shown use an OptiTrack motion capture system.

to a "True" device—n_{i^*}—and $N - 1$ will belong to "False" devices. Thus, we label a feature $\boldsymbol{f}_i$ as 1 if it is estimated to be the True device and 0 otherwise.

9.4 Classification Methods

In order to evaluate the gesture-based IoT selection system, we perform a series of points using a prototype UWB "smartwatch," with infrared reflectors for use with an OptiTrack motion capture system. We use the motion capture data to detect pointing events, which we perform in groups of 5 to ensure accurate ground truth detection. The motion capture provides sub-millimeter position estimates at 60 Hz throughout the duration of each pointing event, allowing

us to accurately tell when and to which device a user is pointing.

An example of these ground truth pointing events is shown in Figure 9.7, where red squares indicate successfully detected pointing groups. Note that there are likely missed pointing events—this occurs most often when the ground truth data is poor, and so pointing events are ignored so as to not contaminate the testing results.

From each feature vector we can estimate whether a particular IoT device is selected or not, but this approach ignores any potential communication or collaboration between connected IoT devices. We introduce two potential classification modes, described below.

Non-collaborative Classification: In non-collaborative classification, each IoT device operates independently, attempting to classify itself as selected or unselected. Other than the communication with n_u required for range estimates, no additional communication is performed. This saves power, but it comes at the cost of very high selection errors.

Collaborative Classification: In collaborative classification, devices are allowed to communicate. Rather than sending entire feature vectors, however, each device will send only a summary of their classification results indicating the certainty with which the classification was made. This allows the network to arrive at a consensus of maximum certainty for which device was selected, allowing for collaboration without prohibitively high communication overheads.

9.4.1 Results

We performed classification for non-collaborative and collaborative schemes using a support vector machine (SVM) and random forest (RF). Note that both of these classifiers can be implemented in embedded hardware quite easily, requiring additions and multiplications in the case of linear SVM (or slightly more complex mathematical operators for alternate kernels) and a series of conditionals in the case of RF. The results for both non-collaborative and collaborative classifiers (as discussed above) are shown in Table 9.1 for SVM using linear, radial basis functions (rbf), and polynomal kernels as well as random forest classifiers.

In non-collaborative classifiers, we attempt to independently classify each device as having

Table 9.1: Classification results for gesture-based IoT device selection, using both non-collaborative and collaborative techniques.

Non-collaborative Classifier	**Precision (%)**	**Recall (%)**	**F-Score (%)**
Support Vector Machine (linear)	39.5	91.7	55
Support Vector Machine (rbf)	52.7	90.5	65
Support Vector Machine (poly)	64.7	85.8	71.2
Random Forest	59.4	93.2	68.6
Collaborative Classifier	**Accuracy (%)**		
Voting on SVM (poly)	88		
Voting on RF	93		

been 'selected' or not selected. In this case, classification accuracy is quite low, with polynomial kernel-based SVM and random forest classifiers performing the best, with high recall percentages (around 85-93%) but a very low precision, at around 60-65%. Note here that precision and recall are defined with respect to true positives (tp), false positives (fp), and false negatives (fn). Specifically, we have $precision = tp/(tp + fp)$ and $recall = tp/(tp + fn)$. Intuitively, precision represents the percentage of events that are reported as positive events and are indeed positive. Recall, on the other hand, represents the percentage of total positive events that are correctly classified. F score is defined as $F = 2\frac{precision \cdot recall}{precision + recall}$, so that $F = 100\%$ represents a perfect classifier.

Collaborative classifiers, on the other hand, have a greatly improved accuracy. This is due to the voting scheme between all N IoT devices done subsequently to non-collaborative classification–we now communicate classification certainties between all devices, arriving at a maximally certain positive label. In particular, in the case of SVM we communicate the margin between a datapoint and its n-dimensional polytope. In the case of RF, we communicate the numerical average of the ensemble prediction. In doing so, we achieve an accuracy of 88% for collaborative polynomial SVM, and 93% for collaborative RF. Here we report only *accuracy*, because at any given pointing event we *know* that only a single IoT device out of N possible is selected. The accuracy is the percentage with which we will choose the correct device. Recall that this 93% is in a 9×10 meter room with 8 devices deployed, or almost 1 device for every 12 m^2 (or 3.5×3.5 m square), which is a very dense deployment.

9.4.2 Comparison with Related Work

In order for the proposed architecture to be viable as a technology for selecting and controlling IoT devices, we must demonstrate that it is scalable and accurate, particularly in comparison with existing or recently proposed methods. Given the deployment density of wireless devices in our lab, we can support a pointing resolution of roughly 1.5 m between devices. In other words, if we had a dense deployment of controllable LED lighting in a smart home setting, these fixtures would have to be spaced apart by 1.5m in order for our system to disambiguate them from pointing gestures made 10m away.

As a point of comparison, WiFi-based 3D tracking and pointing recognition like the work demonstrated by Fadel Adib *et al.* [AKK14] demonstrates 1 m positioning errors and a pointing direction accuracy of 10°. While independently these two results are quite impressive, this resolution implies that at a distance of 10 m, pointing-based device selection would incur $\pm 1m \pm 10\tan(10^{\circ})$, or a swing from -2.76 m to 2.76 m—much greater than our demonstrated results in this chapter. On the other hand, a more accurate result can be achieved using depth sensors like those used in the Microsoft Kinect and Leap Motion. For example, Jing *et al.* demonstrate a device selection with a resolution of 0.6 m using the Kinect [JY]. This, however, requires the user to be within view of a Kinect unit (with a range of about 3.5 m), and it requires multiple, expensive sensors deployed at various intervals throughout the home or office–prohibitively expensive in most scenarios.

The work most closely related to the methods presented in this chapter is that of Mayton *et al.* in developing "WristQue" [MZA13]. WristQue is a wearable watch-like device that contains a magnetic field sensor in order to detect pointing directions based on pre-measured and modeled magnetic field maps for specified indoor environments. Based on detected pointing directions and installed UWB infrastructure that allows for full trilateration-based positioning, [MZA13] provides a similar 1.5 m accuracy in device selection resolution. Note, however, that this method requires painstaking modeling of each room's magnetic field (which could potentially vary over time) and depends on trilateration of ultra-wideband ranges–something that necessitates three or more UWB devices. The method proposed in this chapter can work

for as few as one device, and requires no modeling or pre-calibration prior to operation.

9.4.3 Power Analysis

This device selection technique targets IoT devices that are potentially battery powered. Because of this, care has to be taken to ensure that power consumption is minimized. Here we briefly analyze power consumption for just the mobile (e.g. smartwatch) device that the user carries. Note that in practice, some of the stationary (anchor) IoT devices may be battery powered as well, and in these cases a low power *sniffing* strategy should be employed for listening to messages from the mobile device. For the mobile case, we will ignore the energy cost of computation for classification, as this is dwarfed by the energy required to transmit and receive using the DW1000 UWB transceivers.

In the idle case (when the smart watch is not ranging and is idle, listening for a gesture), $6\mu W$ of power are consumed. When transmitting or receiving, there is a 5ms wakeup period during which $3mW$ of power are consumed, followed by a 260 mW for $200\mu s$ for TX and 370 mW for RX. We will ignore the sleep power consumption (6 μW) for now, as it will be overshadowed by the processing consumption and analog conversions for gesture detection on the smart watch. These power numbers are derived from [dw1]. For N IoT devices, energy consumption during each "pointing" session is calculated as follows:

$$\begin{aligned} E &= K \cdot N^*(E_{wake} + 2 \cdot (E_{TX} + T_{RX} P_{RX})) \\ &= K \cdot N^*(15\mu J + 2 \cdot (52\mu J + 370\mu J)) \end{aligned} \tag{9.3}$$

with a listen period of $T_{RX} = 1ms$ following each transmit (and expected response). Here N^* represents all nodes within communication range, rather than the full number of networked devices, and K is the number of range measurements calculated per node—roughly 20, in the above experiments. For $N = 8$, as is the case in the results shown here, we have $E = 137.4mJ$ per point. For today's smart wearables, a battery with between 750 and 1400 mWh is common—this gives between $2.7kJ$ and $5kJ$ in each battery, meaning that if, for example, 100

pointing gestures are made each day, there is a 0.27% to 0.5% overhead in battery life. This is a very reasonable price to pay for the added convenience of high fidelity gesture-based IoT device selection. With improvements in commodity UWB hardware, however, it is likely that this overhead will decrease further.

9.5 Final thoughts

This chapter demonstrated a departure from the graph realization approaches of previous chapters, though it built upon some of the results from Chapter 8. This chapter offers a final scenario underscoring the importance of joint optimization in mobile localization networks. In this case, joint classification allows for dramatic increases in IoT device selection based on user gestures (or "pointing," to be precise). This can be seen as a reduction in user overhead—that is, the burden placed on users to control an increasing quantity of connected devices.

CHAPTER 10

Discussion

This work demonstrates various advantages to *joint* optimization over a variety of sensing and estimation subsystems in mobile devices. More precisely, we made the claim that sensor fusion and graph realization techniques offer an elegant framework for constraining localization problems with less expensive (in terms of power or cost) sensors, thereby decreasing the overheads commonly associated with precise positioning schemes.

We began by demonstrating work in both RF (Bluetooth) signal strength based localization and RF time-of-flight localization. This served to underscore the weaknesses present in both, motivating the need for fusion techniques to unite the strengths of these techniques and minimize their overheads. For example, while signal strength methods have poor ranging accuracy, they are inexpensive and scalable. On the other hand, time-of-flight techniques have greatly improved accuracy at the cost of increased cost and increased wireless congestion.

We then continued to discuss graph realization techniques (specifically nonlinear least squares GRP solvers) as they pertain to (i) inertial-based path estimation refinement, and (ii) model-training for RF fingerprint localization. The former demonstrated a method for constraining error-prone inertial dead reckoning techniques by incorporating "encounters" between one or more users at various points in time, refining historical position estimates for all users based on GRP that acknowledged relative angles, distances, and positions of user estimates across time. The latter demonstrated how, similarly, GRP formulations served to use inertial path estimates as a means to constrain RF samples taken at various points in an indoor environment. This allowed for decreased overheads (in terms of model training time) when constructing Gaussian process maps for subsequent use in RF fingerprint-based localization. This particular work was

demonstrated in a Microsoft indoor localization competition in Seattle, Washington.

Chapter 6 demonstrated an additional application of joint optimization with inexpensive sensors in the *outdoor* localization setting. Here, when presented with very coarse geolocation information—e.g. from IP mapping—we were able to use pressure sensors to predict driving patterns with accuracy within one city block. This demonstrated an extreme point of "cooperative" localization across the sensing stack—localization estimates may still be obtained even when power consumption is cut by several orders of magnitude. For applications where very rough estimates are adequate—city scale traffic analytics, providing contexts for social services, etc.—this provides a much improved solution for low power positioning services.

Returning to the indoor setting, we discussed in Chapter 8 joint optimization of position and *time*—a measurement that can be thought of as a sensor in an of itself, prone to measurement errors such as offset, drift, and noise. We referred to this work as SLATS, or simultaneous localization and time synchronization. The driving idea was that the intimate relationship between time and range measurements could be used to provide accurate position estimates at decreased cost in terms of power and RF congestion. In particular, while time-of-flight range measurements based on ultra-wideband RF commonly require 3-way messaging schemes, we demonstrated that 2-way or even one-way messaging can be used to provide accurate indoor positioning by fusing accurate clock models learned jointly with position estimates. We demonstrated these results using custom RF expansion boards for commercial quadrotor drones as well as for pedestrian tracking systems.

Finally, we demonstrated how ultra-wideband range measurements can be used to provide an accurate, scalable, and low-power method of wireless IoT device selection based on users "pointing" towards desired devices while wearing compatible devices on their wrist (e.g. a smartwatch). We demonstrated 93% device selection accuracy even with high device deployment density.

In performing the research described above we realized the need for infrastructure that facilitates quick and easy deployment of custom wireless devices, including providing power, wireless (and wired) connectivity, power measurement, and debugging. We developed a cus-

tom hardware solution in an attempt to ease deployment burden. Our solution—NTB or the Networked Test Bed—is intended as an open-source contribution for those requiring quick prototype deployments as well as remotely controllable power, lighting, and measurements. This test bed is currently in use with several additional projects at UCLA outside the scope of this dissertation.

We believe this work demonstrates several valuable contributions to the community of indoor localization, and it opens up opportunities for future work as well, as discussed below.

10.1 Future Work

We leave several potential extensions of the work presented in this dissertation as future work. These include:

SLATS for Distributed Networks: The simultaneous localization and time synchronization architecture presented in this work requires a centralized Kalman filter state estimator. While this is quite practical for settings including small scale homes or offices or high fidelity robotics localization systems, it does not scale well to large, multi-hop networks where communication to a centralized entity may be limited or resource-constrained. One extension to this work would be that of exploring distributed approaches towards fusing time synchronization and localization or ranging estimates.

Single-anchor UWB Localization: The mobile UWB tag (or watch, as we call it) demonstrated in several chapters of this work allow for accurate indoor localization of mobile devices. This watch also contains an inertial measurement unit, and a natural extension of this work could include fusing inertial-measurement-based path estimates with sparsely available UWB range estimates given only a single anchored device in an indoor setting. Such a system would demonstrate an incredibly scalable and practical localization system for home and office environments.

IoT Device Control: We have demonstrated how ultra-wideband can be combined with pattern recognition classifiers to accurately identify a connected device that a user would like

to communicate with. Future work could easily extend this work to create a fully fledged architecture, allowing an "alphabet" of gestures that can be used for general devices upon a user having selected a given device.

APPENDIX A

Time-of-flight Ranging Equations

In deriving equations for single- and double-sided two way ranging, we reference variable names in Figure 8.6 and omit time index k. We use the shorthand $[t_x]_j^i$ to denote time t_x initially measured with respect to node n_i's local clock but converted to node n_j's local clock. Additionally, $[t_x]^i$ is equivalent to $[t_x]_i^i$ and, when unspecified, t_x is equivalent to $[t_x]_j^j$.

Single-sided two-way ranging, r_{ij}:

$$
\begin{aligned}
T_{RND1} &= t_5 - t_2 \\
&= (t_2 + [T_{RSP1}]_j^i + 2T_p) - t_2 \\
&= 2T_p + [T_{RSP1}]_j^i \\
T_p &= (T_{RND1} - [T_{RSP1}]_j^i)/2 \\
\|\boldsymbol{p}_i^* - \boldsymbol{p}_j^*\|_2 &= \frac{c}{2}(T_{RND1} - [T_{RSP1}]_j^i) \\
&= \frac{c}{2}(T_{RND1} - [T_{RSP1}]^i(1 + b_j - b_i)) \\
&\boxed{\approx \frac{c}{2}(T_{RND1} - [T_{RSP1}]^i) = r_{ij}} \\
r_{ij} &= \frac{c}{2}(T_{RND1} - [T_{RSP1}]^i) \\
&\boxed{= \|\boldsymbol{p}_i^* - \boldsymbol{p}_j^*\|_2 + \frac{c}{2}[T_{RSP1}]^i(b_j - b_i)}
\end{aligned}
$$

Symmetric Double-sided two-way ranging, R_{ij}^s:

$$
\begin{aligned}
T_{RND1} &= 2T_p - [T_{RSP1}]_j^i \\
[T_{RND0}]^i &= 2T_p - [T_{RSP0}]_i^j \\
4T_p &= T_{RND1} + [T_{RND0}]^i + [T_{RSP1}]_j^i + [T_{RSP0}]_i^j \\
\|\boldsymbol{p}_i^* - \boldsymbol{p}_j^*\|_2 &= \frac{c}{4}(T_{RND1} + [T_{RND0}]^i + [T_{RSP1}]_j^i + [T_{RSP0}]_i^j) \\
&\boxed{\approx \frac{c}{4}(T_{RND1} + [T_{RND0}]^i + [T_{RSP1}]^i + [T_{RSP0}]) := R_{ij}^s}
\end{aligned}
$$

Asymmetric Double-sided two-way ranging, R_{ij}:

Asymmetric double-sided two-way ranging can be derived by arithmetic manipulation of propagation delay calculations shown below. This has added robustness when $T_{RSP0} \neq T_{RSP1}$ as compared to the symmetric two-way ranging above.

$$
\begin{aligned}
\|\boldsymbol{p}_i^* - \boldsymbol{p}_j^*\|_2 &= cT_p \\
&= c\frac{T_p \cdot 2(T_p + [T_{RSP1}]_j^i + T_{RSP0})}{2(T_p + [T_{RSP1}]_j^i + T_{RSP0})} \\
&= c\frac{4T_P^2 + 2T_p[T_{RSP1}]_j^i + 2T_P T_{RSP0}}{2T_P + 2T_{RSP0} + 2[T_{RSP1}]_j^i} \\
&= c\frac{(2T_P + T_{RSP0})(2T_P + [T_{RSP1}]_j^i) - T_{RSP0}[T_{RSP1}]_j^i}{(2T_P + T_{RSP0}) + (2T_P + [T_{RSP1}]_j^i) + T_{RSP0} + [T_{RSP1}]_j^i} \\
&\boxed{\approx c\frac{[T_{RND0}]^i T_{RND1} - T_{RSP0}[T_{RSP1}]]^i}{[T_{RND0}]^i + T_{RND1} + T_{RSP0} + [T_{RSP1}]^i} := R_{ij}}
\end{aligned}
$$

REFERENCES

[AA04] D. Asonov and R. Agrawal. "Keyboard acoustic emanations." In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pp. 3–11, 2004.

[ABG10] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. "Occupancy-driven energy management for smart building automation." In *Proc. of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pp. 1–6. ACM, 2010.

[AGY04] James Aspnes, David Goldenberg, and Yang Richard Yang. "On the computational complexity of sensor network localization." In *Algorithmic aspects of wireless sensor networks*, pp. 32–44. Springer, 2004.

[AH13] Yuvraj Agarwal and Malcolm Hall. "ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing." In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pp. 97–110, 2013.

[AK13] Fadel Adib and Dina Katabi. "See Through Walls with WiFi!" In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pp. 75–86, New York, NY, USA, 2013. ACM.

[AKK14] Fadel Adib, Zachary Kabelac, Dina Katabi, and Robert C. Miller. "3D Tracking via Body Radio Reflections." In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pp. 317–329, Berkeley, CA, USA, 2014. USENIX Association.

[AR12] M. Angermann and P. Robertson. "FootSLAM: Pedestrian Simultaneous Localization and Mapping Without Exteroceptive Sensors." *Proc. of the IEEE*, **100**(Special Centennial Issue):1840–1848, 2012.

[ASB12] Adam J. Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M. Smith. "Practicality of Accelerometer Side Channels on Smartphones." In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pp. 41–50, 2012.

[AY12] M. Alzantot and M. Youssef. "CrowdInside: Automatic Construction of Indoor Floorplans." *arXiv preprint arXiv:1209.3794*, 2012.

[BBB14] Kevin Bouchard, Abdenour Bouzouane, and Bruno Bouchard. "Gesture Recognition in Smart Home Using Passive RFID Technology." In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '14, pp. 12:1–12:8, New York, NY, USA, 2014. ACM.

[BHW12] Andreas Braun, Henning Heggen, and Reiner Wichert. "CapFloor–A Flexible Capacitive Indoor Localization System." In *Evaluating AAL Systems Through Competitive Benchmarking. Indoor Localization and Tracking*, pp. 26–35. Springer, 2012.

[BI04] Ling Bao and Stephen S. Intille. "Activity recognition from user-annotated acceleration data." pp. 1–17. Springer, 2004.

[BIS08] Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, and Martin Vetterli. "The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments." In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pp. 43–56, New York, NY, USA, 2008. ACM.

[BKO10] David Barrera, H Güneş Kayacik, Paul C van Oorschot, and Anil Somayaji. "A methodology for empirical analysis of permission-based security models and its application to android." In *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 73–84. ACM, 2010.

[BLP06] Michael Broxton, Joshua Lifton, and Joseph A Paradiso. "Localization on the pushpin computing sensor network using spectral graph drawing and mesh relaxation." *ACM SIGMOBILE Mobile Computing and Communications Review*, **10**(1):1–12, 2006.

[bmp] "BMP280 Digital Pressure Sensor." `https://ae-bst.resource.bosch.com/media/products/dokumente/bmp280/BST-BMP280-DS001-10.pdf`. Accessed: 2015-03-04.

[BRS11] Alastair R. Beresford, Andrew Rice, Nicholas Skehin, and Ripduman Sohan. "MockDroid: Trading Privacy for Application Functionality on Smartphones." In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, pp. 49–54, 2011.

[BT05] Jonathan Bachrach and Christopher Taylor. "Localization in sensor networks." *Handbook of sensor networks: Algorithms and Architectures*, **1**, 2005.

[BY04] Pratik Biswas and Yinyu Ye. "Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization." In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, pp. 46–54, New York, NY, USA, 2004. ACM.

[CBA10] Ionut Constandache, Xuan Bao, Martin Azizyan, and Romit Roy Choudhury. "Did You See Bob?: Human Localization Using Mobile Phones." In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pp. 149–160, New York, NY, USA, 2010. ACM.

[CC11] Drew Fisher Keng-hao Chang and John Canny. "Ammon: A speech analysis library for analyzing affect, stress, and mental health on mobile phones." *Proceedings of PhoneSense*, **2011**, 2011.

[CLS12] Mihai Cucuringu, Yaron Lipman, and Amit Singer. "Sensor network localization by eigenvector synchronization over the euclidean group." *ACM Trans. Sen. Netw.*, **8**(3):19:1–19:42, August 2012.

[CLV12] Sundeep Prabhakar Chepuri, Geert Leus, and A van der Veen. "Joint localization and clock synchronization for wireless sensor networks." In *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, pp. 1432–1436. IEEE, 2012.

[cra] "Bitcraze CrazyFlie 2.0." `https://www.bitcraze.io/`. Accessed: 2016-03-09.

[CTS07a] Jose Maria Cabero, Fernando De la Torre, Aritz Sanchez, and Iñigo Arizaga. "Indoor People Tracking Based on Dynamic Weighted Multidimensional Scaling." In *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, MSWiM '07, pp. 328–335, New York, NY, USA, 2007. ACM.

[CTS07b] Jose Maria Cabero, Fernando De la Torre, Aritz Sanchez, and Iñigo Arizaga. "Indoor People Tracking Based on Dynamic Weighted Multidimensional Scaling." In *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, MSWiM '07, pp. 328–335, New York, NY, USA, 2007. ACM.

[DAZ15] Satyam Dwivedi, Alessio De Angelis, Dave Zachariah, and Peter Händel. "Joint Ranging and Clock Parameter Estimation by Wireless Round Trip Time Measurements." *CoRR*, **abs/1501.05450**, 2015.

[DCF09] D. Dardari, A. Conti, U. Ferner, A. Giorgetti, and M.Z. Win. "Ranging With Ultrawide Bandwidth Signals in Multipath Environments." *Proceedings of the IEEE*, **97**(2):404–426, Feb 2009.

[DPA06] Benoit Denis, Jean-Benoît Pierrot, and Chadi Abou-Rjeily. "Joint distributed synchronization and positioning in UWB ad hoc networks using TOA." *Microwave Theory and Techniques, IEEE Transactions on*, **54**(4):1896–1911, 2006.

[DT08] Felix Duvallet and Ashley D Tews. "WiFi position estimation in industrial environments using Gaussian processes." In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 2216–2221. IEEE, 2008.

[DTM13] A.A. D'Amico, L. Taponecco, and U. Mengali. "Ultra-Wideband TOA Estimation in the Presence of Clock Frequency Offset." *Wireless Communications, IEEE Transactions on*, **12**(4):1606–1616, April 2013.

[dw1] "DecaWave DW1000 IR-UWB." `http://www.decawave.com/products/dw1000`. Accessed: 2016-01-26.

[EGC10] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. "TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones." In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pp. 1–6, 2010.

[EGE02] Jeremy Elson, Lewis Girod, and Deborah Estrin. "Fine-grained Network Time Synchronization Using Reference Broadcasts." *SIGOPS Oper. Syst. Rev.*, **36**(SI):147–163, December 2002.

[EMW14] Bernhard Etzlinger, Folker Meyer, Henk Wymeersch, Franz Hlawatsch, Gunter Muller, and Andreas Springer. "Cooperative simultaneous localization and synchronization: Toward a low-cost hardware implementation." In *Sensor Array and Multichannel Signal Processing Workshop (SAM), 2014 IEEE 8th*, pp. 33–36. IEEE, 2014.

[Enr08] Patrizia Tavella Enrico Bibbona, Gianna Panfilo. "The Ornstein-Uhlenbeck process as a model of a low pass filtered white noise." In *Metrologia*, Metrologia 45, 2008.

[FFL07] Brian Ferris, Dieter Fox, and Neil Lawrence. "WiFi-SLAM using Gaussian process latent variable models." In *Proc. of the 20th Intl. joint Conf. on Artifical intelligence*, IJCAI'07, pp. 2480–2485, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[Fox05] E. Foxlin. "Pedestrian tracking with shoe-mounted inertial sensors." *Computer Graphics and Applications, IEEE*, **25**(6):38–46, Nov 2005.

[GG03] Fredrik Gustafsson and Fredrik Gunnarsson. "Positioning using time-difference of arrival measurements." In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 6, pp. VI–553. IEEE, 2003.

[GGS04] Jan Giebel, Darin M Gavrila, and Christoph Schnörr. "A bayesian framework for multi-cue 3d object tracking." In *Computer Vision-ECCV 2004*, pp. 241–252. Springer, 2004.

[GGS13] Mohammad Reza Gholami, Sinan Gezici, and Erik G Strom. "TDOA based positioning in the presence of unknown clock skew." *Communications, IEEE Transactions on*, **61**(6):2522–2534, 2013.

[GKS03] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. "Timing-sync Protocol for Sensor Networks." In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pp. 138–149, New York, NY, USA, 2003. ACM.

[GM13] A. Galov and A. Moschevikin. "Bayesian filters for ToF and RSS measurements for indoor positioning of a mobile object." In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pp. 1–8, Oct 2013.

[Goo15] Google. "Google Elevation API." `https://developers.google.com/maps/documentation/elevation/`, 2015. [Online; accessed 7-March-2015].

[GPL10] Santanu Guha, Kurt Plarre, Daniel Lissner, Somnath Mitra, Bhagavathy Krishna, Prabal Dutta, and Santosh Kumar. "AutoWitness: Locating and Tracking Stolen Property While Tolerating GPS and Radio Outages." In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pp. 29–42, New York, NY, USA, 2010. ACM.

[GST13] Daniel Genkin, Adi Shamir, and Eran Tromer. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis." Cryptology ePrint Archive, Report 2013/857, 2013.

[HBS14] Sebastian Hilsenbeck, Dmytro Bobkov, Georg Schroth, Robert Huitl, and Eckehard Steinbach. "Graph-based Data Fusion of Pedometer and WiFi Measurements for Mobile Indoor Positioning." In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pp. 147–158, New York, NY, USA, 2014. ACM.

[Hen92] Bruce Hendrickson. "Conditions for unique graph realizations." *SIAM J. Comput.*, **21**(1):65–84, February 1992.

[HMS02] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem." In *Distributed autonomous robotic systems 5*, pp. 299–308. Springer, 2002.

[HNT13] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. "Accelerometer-based Transportation Mode Detection on Smartphones." In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pp. 13:1–13:14, New York, NY, USA, 2013. ACM.

[HON12] Jun Han, E. Owusu, L.T. Nguyen, A. Perrig, and J. Zhang. "ACComplice: Location inference using accelerometers on smartphones." In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–9, Jan 2012.

[Ind] IndoorAtlas. "Indoor Positioning using Magnetic Anomalies." www.indooratlas.com/.

[ips] "ipShield: A Framework For Enforcing Context-Aware Privacy." `http://tinyurl.com/ipshieldgit`.

[JGC13] Junghyun Jun, Yu Gu, Long Cheng, Banghui Lu, Jun Sun, Ting Zhu, and Jianwei Niu. "Social-Loc: Improving Indoor Localization with Social Sensing." In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pp. 14:1–14:14, New York, NY, USA, 2013. ACM.

[JJ05] Bill Jackson and Tibor Jordán. "Connected rigidity matroids and unique realizations of graphs." *Journal of Combinatorial Theory, Series B*, **94**(1):1–29, 2005.

[JMV11] Jinseong Jeon, Kristopher K Micinski, Jeffrey A Vaughan, Nikhilesh Reddy, Yixin Zhu, Jeffrey S Foster, and Todd Millstein. "Dr. Android and Mr. Hide: Fine-grained security policies on unmodified Android." 2011.

[JY] Pan Jing and Guan Ye-peng. "Human-computer Interaction using Pointing Gesture based on an Adaptive Virtual Touch Screen.".

[kin] "Microsoft Kinect." `https://developer.microsoft.com/en-us/windows/kinect`.

[KJB15] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. "Spotfi: Decimeter level localization using wifi." In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 269–282. ACM, 2015.

[KKF10] Been Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. "Multiple relative pose graphs for robust cooperative mapping." In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3185–3192, May 2010.

[Kor03] Yehuda Koren. "On spectral graph drawing." In *Computing and Combinatorics*, pp. 496–508. Springer, 2003.

[KPD15] Benjamin Kempke, Pat Pannuto, and Prabal Dutta. "PolyPoint: Guiding Indoor Quadrotors with Ultra-Wideband Localization." In *Proceedings of the 2Nd International Workshop on Hot Topics in Wireless*, HotWireless '15, pp. 16–20, New York, NY, USA, 2015. ACM.

[KPH14] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. "Luxapose: Indoor Positioning with Mobile Phones and Visible Light." In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pp. 447–458, New York, NY, USA, 2014. ACM.

[KPN96] Richard Knoblauch, Martin Pietrucha, and Marsha Nitzburg. "Field studies of pedestrian walking speed and start-up time." *Transportation Research Record: Journal of the Transportation Research Board*, (1538):27–38, 1996.

[LDB07] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. "Survey of Wireless Indoor Positioning Techniques and Systems." *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **37**(6):1067–1080, Nov 2007.

[LFR12] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T Chittaranjan, Andrew T Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. "StressSense: Detecting stress in unconstrained acoustic environments using smartphones." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, Ubicomp '12, pp. 351–360, 2012.

[LHD15] Anton Ledergerber, Michael Hamer, and Raffaello D'Andrea. "A robot self-localization system using one-way ultra-wideband communication." In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 3131–3137, Sept 2015.

[LR12] Patrick Lazik and Anthony Rowe. "Indoor Pseudo-ranging of Mobile Devices Using Ultrasonic Chirps." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pp. 99–112, New York, NY, USA, 2012. ACM.

[LWG13] Sangmin Lee, Edmund L. Wong, Deepak Goel, Mike Dahlin, and Vitaly Shmatikov. "πBox: A Platform for Privacy-preserving Apps." In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, NSDI'13, pp. 501–514, 2013.

[MA07] D. Macagnano and G.T.F. de Abreu. "Tracking Multiple Dynamic Targets with Multidimensional Scaling." In *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, pp. 1–5, Sept 2007.

[MFA07] Guoqiang Mao, Barış Fidan, and Brian D. O. Anderson. "Wireless Sensor Network Localization Techniques." *Comput. Netw.*, **51**(10):2529–2553, July 2007.

[MHG14] Paul Martin, Bo-Jhang Ho, Nicholas Grupen, Samuel Muñoz, and Mani Srivastava. "An iBeacon Primer for Indoor Localization: Demo Abstract." In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, pp. 190–191, New York, NY, USA, 2014. ACM.

[MKM14] Kartik Muralidharan, Azeem Javed Khan, Archan Misra, Rajesh Krishna Balan, and Sharad Agarwal. "Barometric Phone Sensors: More Hype Than Hope!" In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, HotMobile '14, pp. 12:1–12:6, New York, NY, USA, 2014. ACM.

[MKS04] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. "The Flooding Time Synchronization Protocol." In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pp. 39–49, New York, NY, USA, 2004. ACM.

[MST15] J Medvesek, A Symington, A Trost, and S Hailes. "Gaussian process inference approximation for indoor pedestrian localisation." *Electronics Letters*, **51**(5):417–419, 2015.

[MVB12] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. "Tapprints: Your Finger Taps Have Fingerprints." In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pp. 323–336, New York, NY, USA, 2012. ACM.

[MVC11] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. "(Sp)iPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers." In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pp. 551–562, 2011.

[MZA13] Brian D Mayton, Nan Zhao, Matt Aldrich, Nicholas Gillian, and Joseph A Paradiso. "WristQue: A personal sensor wristband." In *2013 IEEE International Conference on Body Sensor Networks*, pp. 1–6. IEEE, 2013.

[NKZ10] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. "Apex: extending android permission model and enforcement with user-defined runtime constraints." In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp. 328–332. ACM, 2010.

[noa] "National Oceanic and Atmospheric Administration (NOAA) National Climatic Data Center." `http://www.ncdc.noaa.gov`. Accessed: 2015-03-05.

[OHD12] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. "ACCessory: Password Inference Using Accelerometers on Smartphones." In *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*, HotMobile '12, pp. 9:1–9:6, New York, NY, USA, 2012. ACM.

[Ope15] OpenStreetMap. "Main Page — OpenStreetMap Wiki,." `http://wiki.openstreetmap.org`, 2015. [Online; accessed 7-March-2015].

[opt] "NaturalPoint (DBA Optitrack) Motion Capture." `https://www.optitrack.com/`.

[OTD04] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. "A boosted particle filter: Multitarget detection and tracking." In *Computer Vision-ECCV 2004*, pp. 28–39. Springer, 2004.

[PAK05] N. Patwari, J.N. Ash, S. Kyperountas, A.O. Hero, R.L. Moses, and N.S. Correal. "Locating the nodes: cooperative localization in wireless sensor networks." *Signal Processing Magazine, IEEE*, **22**(4):54–69, July 2005.

[PCB00] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. "The Cricket Location-support System." In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pp. 32–43, New York, NY, USA, 2000. ACM.

[PPC12] Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. "Online pose classification and walking speed estimation using handheld devices." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pp. 113–122, 2012.

[PRH11] K. Plarre, A. Raij, S.M. Hossain, A.A. Ali, M. Nakajima, M. Al'absi, E. Ertin, T. Kamarck, S. Kumar, M. Scott, Daniel Siewiorek, A. Smailagic, and L.E. Wittmers. "Continuous inference of psychological stress from sensory measurements collected in the natural environment." In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pp. 97–108, 2011.

[QCG09] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. "ROS: an open-source Robot Operating System." In *ICRA Workshop on Open Source Software*, 2009.

[Res] Pew Research. "Cell phone ownership hits 91% of adults." `http://www.pewresearch.org/fact-tank/2013/06/06/cell-phone-ownership-hits-91-of-adults/`. Accessed: 2015-06-24.

[RES10] Sasank Reddy, Deborah Estrin, and Mani Srivastava. "Recruitment framework for participatory sensing data collections." In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive'10, pp. 138–155, Berlin, Heidelberg, 2010. Springer-Verlag.

[RLR14] Niranjini Rajagopal, Patrick Lazik, and Anthony Rowe. "Visual Light Landmarks for Mobile Devices." In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, pp. 249–260, Piscataway, NJ, USA, 2014. IEEE Press.

[RM09] C. Rohrig and M. Muller. "Indoor location tracking in non-line-of-sight environments using a IEEE 802.15.4a wireless network." In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 552–557, Oct 2009.

[Say11] Ali H Sayed. *Adaptive filters*. John Wiley & Sons, 2011.

[SH06] Affan Syed and John Heidemann. "Time Synchronization for High Latency Acoustic Networks." In *Proceedings of the IEEE Infocom*, p. to appear, Barcelona, Spain, April 2006. IEEE.

[Shi09] Katie Shilton. "Four billion little brothers?: privacy, mobile phones, and ubiquitous data collection." *Commun. ACM*, **52**(11):48–53, November 2009.

[Sin11] A. Singer. "Angular synchronization by eigenvectors and semidefinite programming." *Applied and Computational Harmonic Analysis*, **30**(1):20 – 36, 2011.

[SR04] Yi Shang and W. Ruml. "Improved MDS-based localization." In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pp. 2640–2651 vol.4, March 2004.

[ST12] Andrew Symington and Niki Trigoni. "Encounter Based Sensor Tracking." In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '12, pp. 15–24, New York, NY, USA, 2012. ACM.

[STK05] Ali H Sayed, Alireza Tarighat, and Nima Khajehnouri. "Network-based wireless location: challenges faced in developing techniques for accurate wireless location information." *Signal Processing Magazine, IEEE*, **22**(4):24–40, 2005.

[STS13] M. Sousa, A. Techmer, A. Steinhage, C. Lauterbach, and P. Lukowicz. "Human tracking and identification using a sensitive floor and wearable accelerometers." In *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*, pp. 166–171, March 2013.

[SY04] F. Sivrikaya and B. Yener. "Time synchronization in sensor networks: a survey." *Network, IEEE*, **18**(4):45–50, July 2004.

[SZG14] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L. Ananda, Mun Choon Chan, and Li-Shiuan Peh. "Using Mobile Phone Barometer for Low-power Transportation Context Detection." In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pp. 191–205, New York, NY, USA, 2014. ACM.

[TDD11] Stephen P. Tarzia, Peter A. Dinda, Robert P. Dick, and Gokhan Memik. "Indoor Localization Without Infrastructure Using the Acoustic Background Spectrum." In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pp. 155–168, New York, NY, USA, 2011. ACM.

[TM06] Sebastian Thrun and Michael Montemerlo. "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures." *Int. J. Rob. Res.*, **25**(5-6):403–429, May 2006.

[TSW15] J. Tiemann, F. Schweikowski, and C. Wietfeld. "Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments." In *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, pp. 1–7, Oct 2015.

[usa76] *U.S. Standard Atmopshere*. U.S. Government Printing Office, Washington, D.C., 1976.

[VGM15] Leandro A. Villas, Daniel L. Guidoni, Guilherme Maia, Richard W. Pazzi, Jó Ueyama, and Antonio A. Loureiro. "An Energy Efficient Joint Localization and Synchronization Solution for Wireless Sensor Networks Using Unmanned Aerial Vehicle." *Wirel. Netw.*, **21**(2):485–498, February 2015.

[vic] "Vicon Motion Capture." `http://www.vicon.com/`.

[Vin68] T.K. Vintsyuk. "Speech discrimination by dynamic programming." *Cybernetics*, **4**(1):52–57, 1968.

[WLW09] H. Wymeersch, J. Lien, and M.Z. Win. "Cooperative Localization in Wireless Networks." *Proceedings of the IEEE*, **97**(2):427–450, Feb 2009.

[WRD16] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. "Serendipity: Finger Gesture Recognition Using an Off-the-Shelf Smartwatch." In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 3847–3851, New York, NY, USA, 2016. ACM.

[XFM13] Chenren Xu, Bernhard Firner, Robert S. Moore, Yanyong Zhang, Wade Trappe, Richard Howard, Feixiong Zhang, and Ning An. "SCPL: Indoor Device-free Multi-subject Counting and Localization Using Radio Signal Strength." In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, pp. 79–90, New York, NY, USA, 2013. ACM.

[YW13] Hsiao-Chieh Yen and Chieh-Chih Wang. "Adapting Gaussian processes for cross-device Wi-Fi localization." In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pp. 1–8, Oct 2013.

[ZDH13] Xiaoyong Zhou, Soteris Demetriou, Dongjing He, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, Carl A. Gunter, and Klara Nahrstedt. "Identity, Location, Disease and More: Inferring Your Secrets from Android Public Resources." In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pp. 1017–1028, New York, NY, USA, 2013. ACM.

[ZLL14] Chi Zhang, Feng Li, Jun Luo, and Ying He. "iLocScan: Harnessing Multipath for Simultaneous Indoor Source Localization and Space Scanning." In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pp. 91–104, New York, NY, USA, 2014. ACM.

[ZN04] Tao Zhao and R. Nevatia. "Tracking multiple humans in crowded environment." In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pp. II–406–II–413 Vol.2, June 2004.

[ZW10] Jun Zheng and Yik-Chung Wu. "Joint time synchronization and localization of an unknown node in wireless sensor networks." *Signal Processing, IEEE Transactions on*, **58**(3):1309–1320, 2010.

[ZW14] P. Zhang and Q. Wang. "On Using the Relative Configuration to Explore Cooperative Localization." *IEEE Transactions on Signal Processing*, **62**(4):968–980, Feb 2014.