Collecting, Analyzing, and Using Fine-Grain Sensor Data with Mobile Platforms

Richard Wang

CMU-CS-16-110

May 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant numbers CNS-0905277 and IIS-1012733, the Office of Naval Research under grant number N00014-09-1-1031, the Defense Advanced Research Projects Agency under grant numbers FA8750-12-2-0291 and FA8721-05-C-0003, the Carnegie Mellon Portugal Program, Google, and Heinz LARC: Living Analytics Research Center.

Keywords: mobile devices, cell phones, autonomous robots, data collection, sensor measurements, active navigation strategies, wireless handoffs, indoor climate measurements, wireless measurements, path identification from motion trajectories, discrete spatio-temporal representations, environmental insights, data-driven algorithms

To my family and friends.

Abstract

We investigate the challenges of collecting sensor measurements with mobile platforms to reveal spatio-temporal insights across our surrounding environments and enable novel data-driven algorithms. Our efforts aim to take advantage of the increasing ubiquity of mobile platforms are already equipped with sensors. Today, sensor measurements are primarily used in limited settings (outdoor GPS) with coarse-grain accuracy (geofencing). The goal of this thesis is to investigate developments towards fine-grain sensor data with centimeter-level accuracy. In particular, we show that fine-grain sensor maps enable novel data-driven algorithms that leverage awareness of surrounding conditions. In our experience, the biggest challenge is not in using sensor data but actually ensuring continuously up-to-date sensor maps. Therefore, this thesis seeks to addresses the key aspects of collecting, analyzing, and using fine-grain sensor measurements.

Mobile platforms have a distinct advantage over existing data collection efforts due to recent developments in localization and navigation that allow a device to capture labeled sensor data. Benefits include lower costs due to reduced dedicated sensor hardware requirements, reduced human effort due to automatic localization, and increased measurement diversity from moving around. In this thesis, our contributions focus specifically on wireless and indoor climate sensors, where we contend with different measurements characteristics. For data collection, we present a data collection framework for both autonomous robots and cell phones that records data from available sensors. We also develop two algorithms to generate paths for mobile platforms to execute that prioritize sensor measurement needs. For data analysis, we present a discrete spatio-temporal representation so that we can extract fine-grain insights. We specifically introduce navigation adjusted grids, which performs spatial decomposition over the reachable workspace. For data usage, we develop data-driven wireless handoff algorithms that use fine-grain wireless maps to ensure accurate and timely wireless handoff decisions.

We demonstrate the value of these efforts with several concrete contributions. First, we use mobile robots to gather sensor measurements over several months across two enterprise environments. Through our analysis efforts, we reveal fine-grain spatio-temporal insights that show wireless conditions (wireless access point coverage and channel allocation) and indoor climate variations (temperature and humidity changes due to HVAC policies). We also show that data-driven wireless handoffs are able to significantly improve wireless performance for our robot that originally faced intermittent wireless connectivity issues while moving. With the increasing pervasiveness and technological developments of mobile devices, this thesis investigates the opportunity to leverage their sensor hardware to better integrate computing technology with our surrounding environments.

Acknowledgments

This thesis is the product of an unforgettable journey assisted by numerous people along the way for whom I cannot express enough gratitude. I can only hope that these efforts have made a tiny dent in our understanding of the world in which we live.

I would like to thank my advisors Manuela Veloso and Srinivasan Seshan for their wisdom, guidance, and support during my time as PhD student. I am fortunate to have worked on a variety of robots including the Nao's, CoBots and CMDragons team despite my having absolutely no previous background. Working in the intersection of robotics and wireless has been an incredible opportunity and I am grateful for the lessons learned from both of you along the way.

I thank my committee members Peter Steenkiste and Dan Lee for helping to guide the development of this thesis and for their invaluable feedback.

The CoBot has been an essential piece of this thesis and I cannot wait for a future where CoBots are everywhere. This unique and amazing platform is only possible due to the immense efforts of Joydeep Biswas, Brian Coltin, Mike Licitra, and Stephanie Rosenthal.

Competing at RoboCup as part of the CMDragons team has been an exhilarating experience. I am proud to have gone through the experience with Manuela Veloso, Joydeep Biswas, Juan Pablo Mendoza, Danny Zhu, Philip Cooksey, and Steven Klee. It is amazing that we are the best in the world at robot soccer! Thanks to Mike Licitra for designing the SSL robots that have been a pleasure to work on.

Robot development requires many areas of expertise and I am thankful to Susana Brandao, Ryan Cahoon, Somchaya Liemhetcharat, and Junyun Tay for helping me get started. I would also like to thank my remaining collaborators and colleagues that have made this an unforgettable experience: Vittorio Perera, Tiago Pereira, Kim Baraka, Guglielmo Gemignani, Rui Silva, Mehdi Samadi, Prashant Reddy, Tekin Mericli, Cetin Mericli, Ravi Shroff, Yilong Zha, Eric Anderson, Matt Mukerjee, Max Korein, Eleanor Avrunin, Heather Knight, Raulcezar Alves, Rick Goldstein, and Devin Schwab.

I would like to thank my graduate school colleagues who have gone through the PhD experience with me: Kevin Chang, Eugene Wang, Hongyi Xin, David Naylor, Alex Beutel, Nico Feltmen, David Witmer, Yuchen Wu, Zack Coker, John Dickerson, John Wright, Dougal Sutherland, Evangelos Papalexakis, Jay Yoon Lee, Yixin Luo, and Elie Krevat.

Finally, special thanks to my family (Hui-May Chang, Bill Wang, and Michael Wang) and fiancee (Yvonne Yu) for their endless support.

Contents

List of Figures

Chapter 1

Introduction

Our indoor environments today are littered with computing devices equipped with various sensors that closely interact with their surroundings. This includes static sensors like temperature and humidity for HVAC systems mounted across our environments, mobile phones that humans carry with them throughout the day, and most recently mobile robots roam around performing tasks for humans. There is an exciting opportunity to take advantage of these devices to reveal detailed insights about how our indoor environments vary over both space and time. We believe that the benefits of such fine-grain insights are not yet fully appreciated due to the challenges of collecting and maintaining up-to-date sensor maps over time. In this thesis, we seek to address the full gamut of challenges that include: gathering sensor measurements with mobile platforms, analyzing the spatially and temporally diverse measurements collected, and then demonstrating how fine-grain sensor maps enable novel data-driven algorithms.

This thesis did not originally set out to focus on data collection; it emerged as we found that data-driven algorithms presented a pragmatic solution for improving the operation of our own autonomous robot as shown in Figure 1.1. We were initially interested in resolving our own autonomous robot's intermittent wireless connectivity issues that arose while moving. Remote telepresence viewers monitoring the robot would lose connectivity for several minutes and be unaware about the state of the robot. Sometimes the robot would reappear on opposite ends of the building and other times it would be stuck behind leaves of a potted plant. We uncovered poor wireless handoffs as the root cause and devised a pragmatic wireless solution that takes advantage of fine-grain wireless maps to reveal surrounding wireless conditions. Our data-driven algorithms significantly improved wireless performance within a self-contained solution on the robot that could be deployed across real wireless networks.

A key assumption of our wireless solution was access to accurate wireless maps; unfortunately, ensuring up-to-date wireless maps became a key challenge during long-term deployment of our data-driven algorithms as measurements become stale when the surrounding environment changes. As a result, we started to focus on data collection and noticed that these challenges are not unique to wireless measurements but also extend to other sensors data like temperature, humidity, air quality, etc. We first realized that our use of an autonomous robot to capture sensor measurements was very powerful as we were leveraging their unique localization and navigation

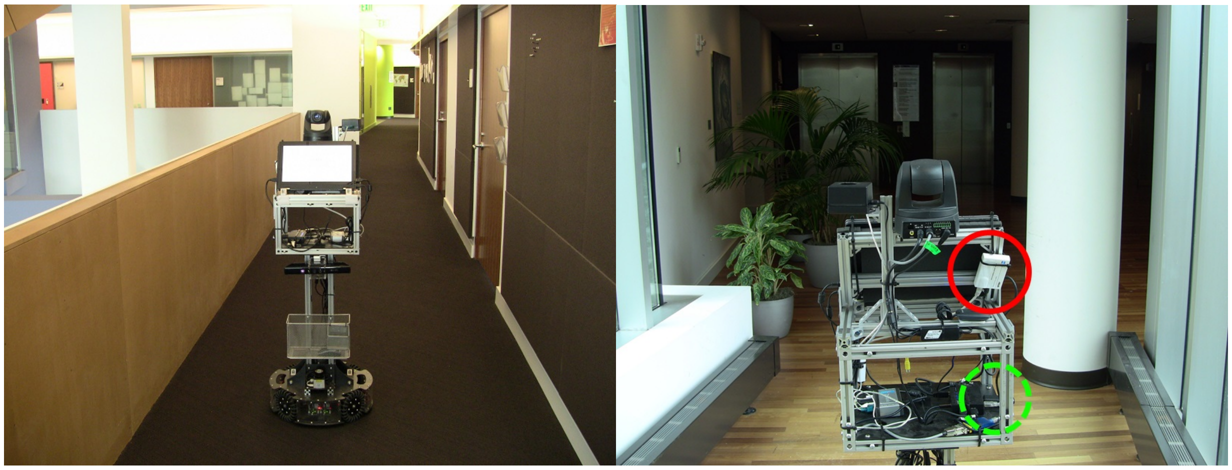

(a) Robot navigating corridors. (b) Robot with sensors mounted.

Figure 1.1: Autonomous robots in our indoor environments can help collect sensor data.

abilities. Of course, simply collecting large numbers of raw sensor measurements has limited value so we directed our efforts to extracting spatio-tepmoral insights from them. As one can see, the pattern of one problem leading to another led to the development of this thesis where we address a wide range of challenges relating to utilizing sensor data opportunistically gathering using mobile platforms.

Let us more concretely define the contributions of this thesis. Data collection with a mobile platform that moves is very different from alternate efforts like static sensor deployments. There are several key challenges that we focus on in this thesis. First, a data collector that moves is trading temporal granularity for spatial granularity: a static sensor continuously collects measurements from one location while a moving sensor collects from numerous locations over time. This affects the entire data collection system with issues that include accurately determining the location of each sensor measurements and quantifying measurements over both space and time. Second, mobile data collection platforms like robots whose motion we can control creates the unique opportunity to direct future measurement collection based on sensor data needs. Finally, we must also determine how to use fine-grain sensor maps not just for extracting insights about our environments but also for enabling algorithms to make better decisions.

We provide several contributions in this thesis to address the challenges of using mobile platforms for collecting sensor measurements. First, we develop a general data collection framework that allows us to effectively recover the spatio-temporal context of sensor measurements collected as the mobile platform *moves* across the environment. In their raw form, such fine-grain measurements provide limited value as the sensor values are dependent on location and time. While some measurements may be highly correlated and differ by as little as a few centimeters and milliseconds, others may be completely uncorrelated if captured on completely different floors or disparate times. Our next step is to process these measurements based on spatio-temporal considerations so that we can more effectively quantify and extract insights from the large number of measurements.

One time data collection efforts are insufficient as most realistic environments operate in continuously changing environments. Ensuring up-to-date sensor maps requires that mobile devices periodically capture measurements from the same location over time. With autonomous robots whose motion we can control, we have a unique opportunity to have the robot to automatically move around and collect measurements based on data needs. Given these finite data collection times, these algorithms need to compute navigation trajectories that effectively use the robot's time.

Previous efforts have shown that sensor measurement maps support applications that include climate control for preserving museum art pieces Michalski [2007], more efficient HVAC control for building management Agarwal et al. [2010], Pérez-Lombard et al. [2008], and management of wireless network configurations Bahl et al. [2005]. While more efficient centralized controllers are valuable, collection of fine-grain sensor maps can also benefit mobile devices operating in the environment by enabling decentralized, context-specific decision making. As an example, we specifically highlight how an autonomous robot overcomes the challenges of intermittent wireless connectivity by simply using a wireless map. This practical solution contrasts traditional efforts that either modify the wireless infrastructure or wireless protocol.

In summary, this thesis focuses on the task of data collection with mobile platforms including both cell phones and autonomous robots. To address the key challenges in using these devices for collecting and maintaining up-to-date sensor maps, our main contributions include a general data collection framework, a discrete spatio-temporal representation to extract insights, an algorithm for generating data collection navigation strategies, and a concrete use case showing that fine-grain sensor maps can help to overcome challenges wireless connectivity issues while moving.

1.1 Thesis Question

This thesis seeks to answer the question,

> With the emergence of sensor-equipped mobile platforms across our indoor environments, how can we collect, maintain, and use insights about fine-grain variations of our surrounding environments?

Sensors found on a wide range of mobile platforms can collect valuable insights about our environments. Making sense of these measurements is difficult because our indoor environments are affected by many factors ranging from building structure to materials as well as building occupants. Ensuring up-to-date sensor maps is also challenging because our environments change over time, which necessitates periodic measurement capture and discarding of stale measurements. Finally, utilizing such fine-grain maps requires identifying problems where mobile platforms in our environment benefit from fine-grain understanding of their surroundings.

In this thesis, we show that the emergence of autonomous robots allows us to reveal fine-grain spatio-temporal insights of our surrounding environments. To mitigate the threat of stale sensor maps, we develop active trajectory generation algorithms that enables these robots to automatically move across the environment based on data collection needs. Finally, we present a concrete example of data-driven algorithms that show how wireless maps can be used to overcome challenging wireless connectivity issues.

1.2 Approach

We approach the thesis question from multiple directions with a focus on the unique data collection challenges that arise with mobility. The common theme across our efforts is the obsession with ensuring accurate and timely fine-grain sensor maps. The four key challenges are: collecting measurements for these maps, creating maps that reveal spatio-temporal trends, generating trajectories to periodically update maps over time, and showing that fine-grain maps lead to algorithms that can overcome challenging problems.

Collecting measurements with mobile platforms like autonomous robots allows us to uniquely capture measurements at many more unique locations than alternate efforts. A unique challenge that arises is accurately estimating the location of each sensor measurement. Fortunately, robots already localize with high accuracy for navigation, which means we simply need to properly match sensor measurements with their corresponding location. In contrast, mobile phones lack these localization abilities but we introduce a map matching approach that is centered around utilizing the navigation map to reason geometrically about where the device must have traversed. We develop a general data collection framework that opportunistically captures measurements from available sensors and then fuses them with continuous localization estimates. Through such efforts, we are able to see fine-grain variations from long-term data collection across actual enterprise environments.

Next, we need to process the raw sensor measurements so that we can reveal fine-grain insights about our surrounding environments. The challenge is that the measurements captured over a nearly continuous space of different locations and times have very different measurement context. While measurement centimeters apart may be highly correlated, those captured weeks apart may be completely uncorrelated. Our approach is to group measurements with similar spatio-temporal context so that we can quantify sensor variations over time and space. We investigate discrete representations of the environment that are specifically tailored for the limited regions that the device can reach. By focusing on the device's navigation map, we are able to better utilize the finite set of sensor measurements collected.

Ensuring up-to-date maps requires revisiting locations over time and a robot whose movements can be controlled allows us to pro-actively direct future measurement capture when opportunities arise. There are two key challenges when pro-actively generating data collection trajectories: effectively using the robot's limited data collection time and adapting to changing measurement needs over time. After all, it is less valuable to capture measurement from a location that was recently visited but it might also be important to focus on locations experiencing more anomalous behavior. We introduce path planning algorithms that that effectively balance the robot's time to navigate between locations with the spatio-temporal value of sensor measurements at different locations. We demonstrate these algorithms effectively utilize the robot's limited deployment time when given opportunities to perform such pro-active data collection efforts.

Finally, we investigate the value of fine-grain sensor maps by showing that we can enable novel data-driven algorithms. For example, we address the problem of intermittent wireless connectivity issues that were plaguing our autonomous robot, which manifested in poor remote telepresence experiences due to frequent interruptions in the remote video stream while moving. We show that fine-grain wireless maps enables highly accurate and timely wireless handoff decisions that resulted in significantly improved wireless performance. While this application is very niche, it shows that sensor maps can enable extremely accurate and timely decisions that will be important as our computing devices become further integrated with the operation of our environments.

1.3 Contributions

The key contributions of this thesis are the following:

Data Collection Framework from Mobile Devices In Chapter 3, we introduce a data collection framework for autonomous robots centered around taking advantage of their ability to localize continuously and navigate autonomously. We match each sensor measurement with their corresponding location, which allows us to collect large numbers of high granularity measurements across many locations. To better understand variations across our environments, Chapter 4 shows how sensor measurements with similar spatio-temporal contexts are grouped together based on a discrete representation centered around regions where the robot can reach. This allows us to reveal variations across our environments and quantify differences in measurement needs over both space and time.

Mobile Phone Trajectory Identification Figuring out where a mobile device has been is necessary to recover locations of captured sensor measurements. Unlike robots that are equipped with powerful exteroceptive sensors, localization with cell phones is much more challenging. In Chapter 6, we develop a robust path identification algorithm that reshapes noisy motion trajectories to the limited navigation trajectories in order to determine where the device must have traversed. We show that this approach can even overcome slightly incorrect maps.

Pro-active Navigation Based on Measurement Needs Sensor values across our environments can change unpredictably over time making it a challenge to ensure up-to-date maps. In Chapter 5, we develop algorithms that opportunistically use robot free time when no higher priority tasks need to be performed by pro-actively generating trajectories to seek out high priority measurement locations that effectively make use of the robot's limited time. We extend our approach to also incorporate the time value of measurements that inevitably arise due to the periodic nature of our surrounding environments.

Use of Fine-Grain Sensor Maps for Timely Decision Making To show the benefit of sensor maps, Chapter 7 shows how we overcome our own robot's intermittent wireless connectivity issues as an example of the type of accurate and timely decisions an agent can make when equipped with such fine-grain maps. After diagnosing the root cause of intermittent connectivity as poor wireless handoffs, we use the wireless map to pre-compute a wireless handoff plan that the robot then uses to seamlessly switch access points in real-time as it traverses wireless handoff waypoints. Using location to trigger wireless decisions contrasts traditional wireless solutions that react to noisy signal strength measurements. This example demonstrates the value of fine-grain sensor maps to enable novel data-driven algorithms that take advantage of knowledge about the surrounding environment.

1.4 Reading Guide

We summarize each chapter below and also provide a visual overview shown in Figure 1.2.

Chapter 2 - Impact of Sensing on Robots We introduce relevant background on indoor mobile devices and highlight how their sensor capabilities differentiate their localization and navigation abilities. We also discuss several types of specific sensors that such mobile devices can be easily equipped with that inform us about our surrounding environments. We present a case study on how robots can benefit by better understanding their surroundings.

Chapter 3 - Data Collection with Autonomous Robots We consider the challenges of using autonomous robots as a data collection platform. We present a data collection framework that allows us to opportunistically collect measurements from a variety of mounted sensors. We discuss how such measurements are effectively merged based on timestamp in order to mark each measurements with their corresponding location. Finally, we highlight the challenges of with these raw sensor measurements.

Chapter 4 - Representation and Processing of Fine-Grain Measurements We group similar measurements based on spatio-temporal contexts so that we can easily quantify measurement differences over time and space. We specifically introduce a discrete representation of the environment that is tailored for mobile data collectors as the region boundaries are based on robot reachable areas. We show detailed insights captured from long-term deployments across two enterprise environments.

Chapter 5 - Active Navigation of Robots for Measurement Needs We generate navigation trajectories based on data collection needs that are targeted as mobile robots with limited data collection opportunities due to interruptions by higher priority tasks. Such trajectories not only consider the variations in measurement value over space but also incorporate the time value of measurements.

Chapter 6 - Data Collection with Other Mobile Devices: Cell Phones While the previous chapters focus on mobile robots, crowd-sourcing sensor measurements from mobile phones is an alternative approach. The biggest challenge for cell phones is robust localization in order to properly match each sensor measurement with their corresponding locations. We present a trajectory snapping approach that reshapes noisy motion trajectories to the limited navigable trajectories in order to recover the path traversed.

Chapter 7 - Use of Sensor Maps: Mobile Wireless Handoffs One of the benefits of fine-grain sensor maps is enabling agents in the environment to make context-specific decisions. As an example, we investigate motion-based wireless connectivity issues that resulted in poor telepresence experiences. We highlight how fine-grain wireless maps resulted in significantly improved wireless handoffs due to better informed wireless decisions based on device context. In contrast to solutions that typically require modification of infrastructure hardware or protocol modifications, our practical solution only modifies the handoff algorithm on the target wireless client.

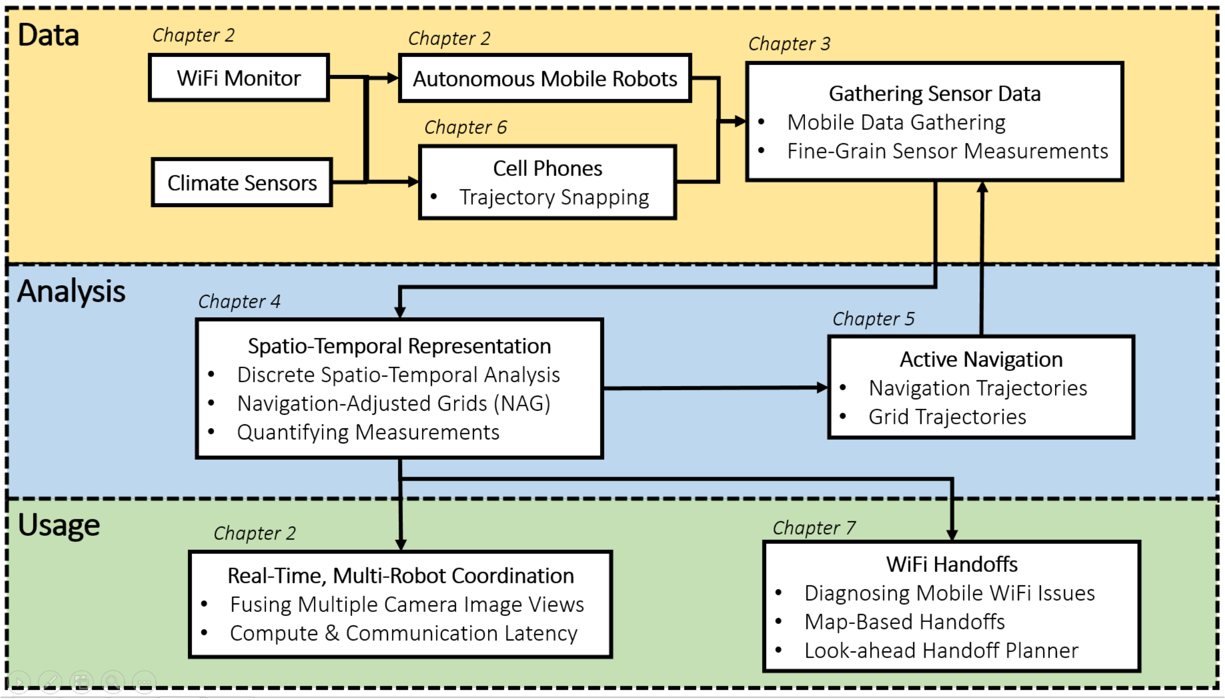

Data
Chapter 2
WiFi Monitor
Climate Sensors
Chapter 2
Autonomous Mobile Robots
Chapter 6
Cell Phones
• Trajectory Snapping
Chapter 3
Gathering Sensor Data
• Mobile Data Gathering
• Fine-Grain Sensor Measurements
Analysis
Chapter 4
Spatio-Temporal Representation
• Discrete Spatio-Temporal Analysis
• Navigation-Adjusted Grids (NAG)
• Quantifying Measurements
Chapter 5
Active Navigation
• Navigation Trajectories
• Grid Trajectories
Usage
Chapter 2
Real-Time, Multi-Robot Coordination
• Fusing Multiple Camera Image Views
• Compute & Communication Latency
Chapter 7
WiFi Handoffs
• Diagnosing Mobile WiFi Issues
• Map-Based Handoffs
• Look-ahead Handoff Planner

Figure 1.2: Overview of contributions by chapter.

Chapter 2

Background on Mobile Platforms and Sensing Devices

Using mobile platforms for data collection differs from traditional efforts that rely on static sensors. First, collecting sensor measurements while moving to different locations requires accurate localization to determine where each sensor measurement was captured. Inherent differences in device capabilities and freedom of movement influences the ability to continuously localize the device. Second, adding sensor hardware to existing mobile platforms requires consideration of the device's existing form factor and power requirements. In addition, specific sensor properties about capture conditions influences the accuracy and reliability of measurements captured by a device that moves.

In this chapter, we first introduce relevant background on the fundamental differences between different types of mobile devices and types of sensor data that can be captured. This naturally leads to differences in robustness and accuracy in their localization abilities, which ultimately affects the challenges in using them for collecting fine-grain sensor measurements. We then describe how different sensors have different measurement characteristics, which are important considerations for our efforts that will gather these measurements with mobile platforms that move around.

Finally, we show why sensor measurements need additional processing in order to be used effectively. We present a concrete example that involves sharing sensor measurements across multiple robots to help make real-time decisions for intercepting a moving ball. This task requires that we identify robust pairs of corresponding sensor measurements, compute how to transform sensor data into the egocentric coordinates of the actuating robot, and fuse these disparate measurements by incorporate computational and communication delays. We hope that these efforts motivate why the contributions of this thesis need to cover all three challenges of data collection, analysis, and usage.

2.1 Mobile Platforms

Mobile platforms today can be separated into the following categories: mobile phones, telepresence robots, and autonomous robots. In order to determine the potential effectiveness of different mobile platforms as opportunistic data collectors, we need to understand their capabilities and typical usage. For example, autonomous robots are highly capable data collectors because they continuously localize with high accuracy and their large form factor enables the addition of various sensor hardware. Given that these robots roam around performing tasks for humans, they are perfectly situated to gather valuable sensor data. In contrast, mobile phones localize poorly and are equipped with a limited set of sensors; however, they are nearly ubiquitous making it desirable to use them as data collectors.

In this section, we define fundamental differences across these mobile platforms that include form factor, hardware limitations, movement constraints, and typical usage. We also discuss how this affects their usage for opportunistic data collection: localization accuracy, sensor hardware capabilities, freedom of movement, and presence across indoor environments. By trying to understand the inherent limitations of different devices, we can develop algorithms to work with their capabilities. For example, Chapter 6 notes that while cell phones lack powerful exteroceptive sensors for localization, we can alternately utilize motion trajectories from motion sensors in order to recover paths traversed.

Mobile Phones

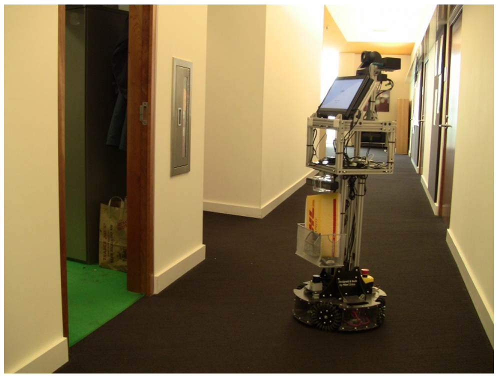
These small, low power devices can be held in a wide range of orientations and can be easily placed in many locations across our environments. Accompanied with their rapid proliferation, these devices are ideal for crowdsourcing data from many unique vantage points. They are often equipped with motion sensors that usually include accelerometers and gyroscopes (IMU), cellular radios, WiFi radios, GPS, cameras, and barometers. Given their small form factor, a small battery means low power consumption is important. The low power sensors generally include the motion sensors, compass, barometers while the higher power consuming sensors include cellular, WiFi, and GPS. The combination of these constraints makes it very difficult for cell phones to have accurate contextual awareness in indoor environments.

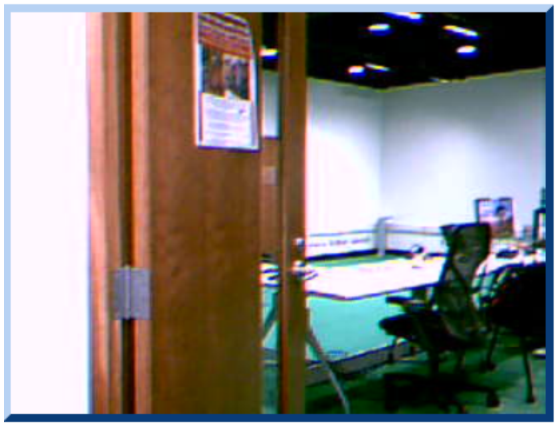
Telepresence Robots

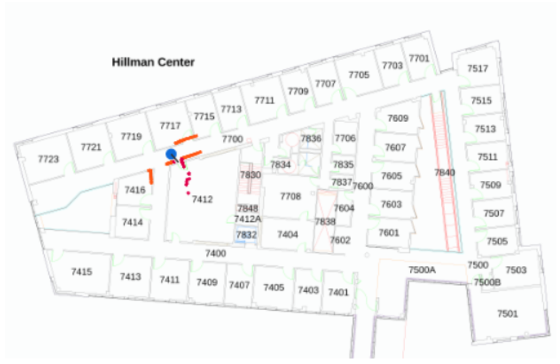
Teleoperated/telepresence robots are laptops on wheels with an external battery but their operation completely depends on remote manual control by humans. The biggest difference with autonomous robots is that they do not require powerful exteroceptive sensors since they only need the robot to perform basic obstacle avoidance. As a result, the operation of teleoperated/telepresence robots are completely dependent on the reliability of WiFi connectivity, which turns out to be a big challenge in practice beam.

Autonomous Robots

As shown in Figure 2.1, autonomous robots are also laptops on wheels but equipped with high capacity external batteries and powerful exteroceptive sensors like Kinect and LIDAR. Autonomous robots today are capable of driving large distances across an environment without human assistance via autonomous navigation and perform various tasks like delivering messages, packages, and escorting visitors Veloso et al. [2012b]. In contrast to cell phones, autonomous robots are large, heavy and often restricted to movements on a 2D plane, which is sufficient for navigating across indoor, office environments. Autonomous robots can operate without WiFi connectivity but wireless communication does help with overcoming lack of knowledge by querying the web Kollar et al. [2012], Samadi et al. [2012], Nardi and Veloso [2013], lack of computation by offloading to the cloud Ventura et al. [2013], and also enables remote telepresence (real-time video in Figure 2.1b and tracking robot status in Figure 2.1c).


(a) Robot delivering package


(b) Video streaming over WiFi



(c) Sensing and location updates over WiFi

Figure 2.1: Autonomous robots perform tasks and use WiFi for remote telepresence.

2.2 Localization and Navigation

Localization is essential for data collection with mobile platforms in order to properly label each sensor measurement. In fact, the accuracy of these labels naturally affects the accuracy of the insights we can extract as will be discussed in Chapter 3. In this section, we provide details about localization efforts to show why it is difficult to robustly track devices as they move and also the accuracy of current efforts. Figure 2.2 reiterates hardware differences so we can better see that exteroceptive sensors are primarily responsible for robots dominating cell phones for localization. It also show why only robots are capable of autonomous navigation due to the presence of controllable wheels.

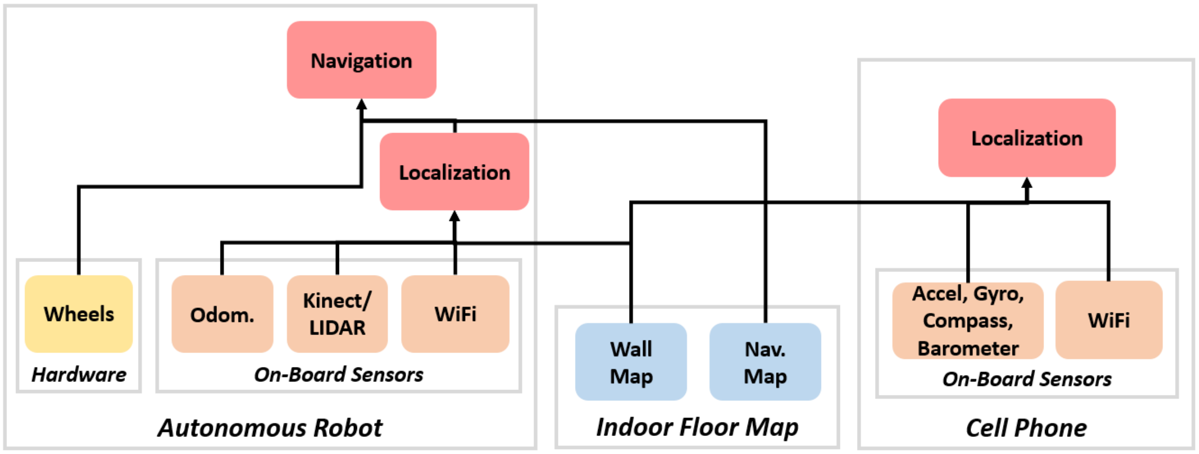


Figure 2.2: Hardware differences between robots and cell phones.

Mobile Phones

Cell phones lack powerful exteroceptive sensors found in robots and have great freedom of movement, which complicated localization. Many efforts have focused on exploiting various on-board sensors found on cell phones that are combined with localization techniques from robots. Unfortunately, cell phone localization remains a challenging problem in realistic environments with most practical solutions requiring significant human overheads.

Dead reckoning efforts with motion sensors have been unsuccessful due to the accumulation of drift over time Kelly [2004], Doh et al. [2003]. Some efforts have looked at use particle filters to correct dead reckoning errors with a known floor plan but are challenged by open spaces and changing floor plans Chintalapudi et al. [2010], Woodman and Harle [2008]. Alternative indoor localization efforts try to exploit globally consistent sensor data. WiFi localization via GPS-like triangulation Bahl and Padmanabhan [2000], Youssef and Agrawala [2005], Lim et al. [2005], Cheng et al. [2005], Serra et al. [2010] have an accuracy of 3m but remains challenging in practice since the unpredictability of signal propagation requires arduous collection of fingerprint maps. Similar challenges are faced with other signals like FM Chen et al. [2012], ambience Azizyan et al. [2009], and acoustic Nandakumar et al. [2012], Liu et al. [2012]. Some approaches combine different types of complementary sensor data to take advantage of high resolution motion updates while correcting drift with coarse, globally consistent measurements Rai et al. [2012], Constandache et al. [2010], Chen et al. [2005].

Many recent efforts have tried to extend SLAM techniques to automatically locate globally unique landmarks that include WiFi fingerprints and magnetic signatures. Some approaches build maps from a combination of magnetometers, WiFi, and accelerometers Wang et al. [2012]. Other approaches notice that RSSI values are not consistent across different wireless devices but that RSSI peaks occur at similar positions. Instead of using all wireless measurements, they select robust RSSI peaks to correct for motion drift errors Shen et al. [2013]. The best efforts have an accuracy of 1.6 meters.

Telepresence Robots

The few sensors telepresence robots have (e.g. RGB cameras, basic proximity sensors) are focused on providing information to assist humans in remotely navigating the robot. These robots do not localize and so they also do not navigate autonomously.

Autonomous Robots

Robot localization is considered a solved problem for static environments. Particle filters have been successful for continuously estimating robot location while moving but require a known floor plan. Particle filters are probabilistic techniques that continuously estimate and track the pose of mobile robots by using known floor maps Fox et al. [1999], Doucet and Johansen [2009]. Particles sample the posterior of robot poses by combining uncertainties from motion and then updating the likelihood of observations from the environment. These efforts can accurately localize within 10 cm by using powerful exteroceptive observations like Kinect and LIDAR. One of the challenges of particle filters is the computational cost of maintaining sufficiently large numbers of particles to accurately estimate robot poses. Subsequent efforts have reduced the computational cost of particle filters to operate robustly in real-time on a tablet Biswas et al. [2011].

Efforts in SLAM have eliminated the need for floor plans by automatically building a map while simultaneously localizing the robot Durrant-Whyte and Bailey [2006], Bailey and Durrant-Whyte [2006]. These efforts estimate the joint state of robot poses and observations of landmarks by solving an optimization problem Thrun and Montemerlo [2006]. SLAM techniques typically work best in stationary environments. Some remaining challenges include long-term SLAM due to growing computational complexity, dynamic environments that change over varying time scales, and use of RGB cameras instead of highly accurate lasers Aulinas et al. [2008].

2.3 Sensing Devices

Every sensor has its own unique sensor measurement considerations. When our data collection efforts move these sensors to different locations, we subject the sensor to different measurement conditions that requires awareness of how this may affect the interpretation of these sensor values. For example, temperature sensors take several minutes to reach equilibrium if the sensor moves across many locations during that time. In this thesis, we focus on wireless signals and indoor climate readings because they provide important insights about our environments that can be re-configured (e.g. wireless network access points can be moved and HVAC can adjust climate policies) and force us to consider sensors with a range of different behaviors (e.g. time to reach equilibrium and sensor parameter choices like wireless channels).

Indoor Climate Indoor digital climate sensors return the measured temperature and relative humidity. Capturing measurements from these sensing devices is straightforward and there are no sensor parameters to consider. We equipped our robots with the AM2303 and AM2315 sensors that are connected to an Arduino UNO that serially publishes messages to our robot. The temperature and relative humidity values are updated at a maximum fixed frequency of 0.5 Hz (1 update every 2 seconds).

802.11 WiFi As nearby wireless devices transmit wireless data, a WiFi monitor captures the received signal strength (RSSI) of each wireless packet. In many environments today, we can capture wireless packets at rates over 300 Hz. To understand the why such measurements are important, we start by describing basic properties of wireless signals transmitted by wireless device, then the importance of access points (APs) for wireless connectivity, and finally explain the difficulty of good wireless performance while roaming.

Unlike wired data that is transmitted over a controlled medium, wireless signals are transmitted over the air and subject to interference, reflections, and attenuation from both materials and other devices in the surrounding environment. In realistic environment, any of these factors could influence wireless signals, which makes it even more difficult to fully understand WiFi performance in practice.

We refer to WiFi devices that wish to connect to the wireless network as *wireless clients*. To connect to enterprise Wifi networks, WiFi clients find and connect to one of the enterprise's WiFi infrastructure *access points* (APs). WiFi APs are statically mounted throughout the building to provide WiFi clients at various locations in the environment with a wireless link to the network. These APs are connected to the enterprise's wired network in order to provide Internet connectivity. Due to their limited range, these APs are carefully placed throughout the environment to ensure that at least one AP is visible at all locations. They are also centrally managed by a controller to ensure that they cooperatively utilize the shared wireless medium.

When WiFi devices move around an environment, AP handoffs are necessary to ensure sustained WiFi connectivity. Since device movement requires AP handoffs, the overheads and inefficiencies of poor AP handoffs accumulates results in significantly degrade WiFi performance.

Scan-based AP handoffs algorithms are used by most mobile devices today because they work ubiquitously across most WiFi networks. It has been shown that poor handoffs can be the reason for degraded WiFi performance while moving Raghavendra et al. [2007]. Previous efforts have considered many approaches for improving AP handoffs that include improved handoff protocols, handoff prediction, and vehicular handoffs.

Acoustic/Audio Acoustic signals have the same propagation properties as wireless signals except they are transmitted over audible frequencies that humans can hear and they are generally not used for data transmission. This means that acoustic signal can also be used for localization with meter level accuracy Nandakumar et al. [2012]; in fact, one can even localize similarly for FM signals as well Chen et al. [2012]. As a sensor, acoustic signals are unique as they can be used for application like speech recognition Benzeghiba et al. [2007] and human robot interaction Fong et al. [2003].

2.4 Benefits of Fine-Grain Measurements

Previous efforts with dense static sensor deployments have shown the value of fine-grain sensor measurements Bahl et al. [2005]. Mobile platforms are not only a pragmatic solution for collecting such measurements but more importantly collect even more fine-grain measurements because they can move across numerous locations over time. We believe that these data collection efforts are mutually beneficial to the mobile platforms collecting the measurements but also to others in the surrounding environment. As an example, we enumerate concrete examples of applications that benefit mobile robot data collectors as well as wireless network administrators.

Wireless experts can use robots to better monitor the difficulties of wireless connectivity in practice:

- **Reveal Environment-Specific Wireless Conditions -** effective management of wireless networks requires understanding wireless signal propagation, which autonomous robots can help to reveal via fine-grain data collection
- **Serve as a Robust Wireless Testbed -** enable repeatable testing and evaluation of different wireless configurations using not only just static wireless devices but also devices that move
- **Investigate Extreme Mobility Challenges -** few other devices push the boundaries of wireless connectivity in terms of continuous streaming of real-time video while moving quickly across an environment

Mobile robots benefit from these same wireless measurements because connectivity is essential for numerous value-add applications (e.g. remote telepresence and cloud-based computing):

- **Differentiate Wireless Hardware Performance -** subject different wireless hardware to repeatable conditions to objectively evaluate wireless performance in practice as opposed to theoretical performance claims posted by manufacturers

- **Identify Wireless Connectivity Issues -** a variety of wireless issues can lead to poor wireless performance in practice making it important to equip robots with tools to identify the root cause

- **Assist with Wireless Network Management -** the wireless infrastructure serves as the backbone of robust wireless performance so helping network administrators to quickly diagnose the wireless network helps both the robots and the admins

As we can see, sensor data collection with mobile platforms create many mutually beneficial applications for both. In these examples, not only are robots pushing the boundaries of wireless connectivity, they have an interest in revealing these wireless challenges so that experts can help to resolve them. More generally, as we continue to see the rise of mobile platforms in our environments, it benefits both the device itself and surrounding devices to better understand context-specific variations across our environments.

2.5 Case Study: Collecting, Sharing, and Using Sensor Measurements

We present a concrete case study to not only show the potential value of sensor measurements but also highlights the challenges of effectively using sensor measurements. We want to show that the reason sensor data is not more widely used today is not because of a lack of trying, it is primarily due to the data processing in order to use sensor data well. For example, this case study focuses on sharing visual information across two robots for a time critical task. Challenges include combining sensor measurements from different devices, contending with stale measurements that no longer apply to current conditions, and understanding the impact of computational and communication delays for a time sensitive application. While the remaining chapters of this thesis focus on sensor maps in particular, similar challenges remain because we need to align sensor measurements along map coordinates, ensure sensor maps are up-to-date, and effectively time when and where these measurements are used. We hope the example presented in this section motivates why this thesis tries to build a complete data collection system touching a range of topics including collection, analysis, and usage.

For this case study, our goal is to help a robot overcome its limited actuation speed and range of vision to more successfully perform the task of intercepting a moving ball by utilizing a teammate's view. Relying on its own vision is insufficient because sensor uncertainty and hardware limitations lead to either untimely or inaccurate decisions that result in failure. Sensor measurements shared are ball observations within each robot's own local frame. To combine these ball observations, we must discover the relative geometric transform between the two robots via correspondences. To ensure reliable and fresh correspondences, we using a switching kalman filter to identify high confidence correspondences. Our empirical results show the importance of effectively using the shared sensor measurements to increase the success of achieving the goal.

Utilizing Teammate Views

We decided to use ball observations alone with the assumption that there are insufficient other identifiable objects within overlapping views of both robots. This is realistic for many natural surfaces like soccer fields where everything is covered with grass. Our approach accumulates correspondence points from simultaneous ball observations over time to overcome the assumption of a sparse object environment. We also assume that the robots remain stationary while accumulating correspondence points. As shown in Figure 2.3, two robots viewing the same object should be able to recover a reasonable relative transform between one another. The problem is that raw measurements are inaccurate due to noise and false ball identifications so we first track the state of the object to detect robust correspondences and then compute the robot's relative transform.

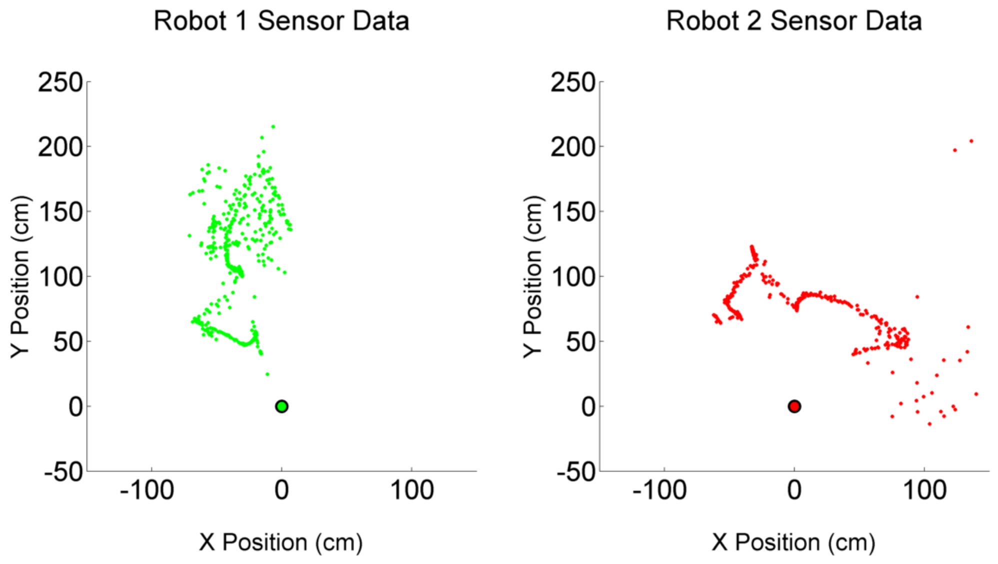


Figure 2.3: Simultaneous ball observations in each robot's egocentric coordinates.

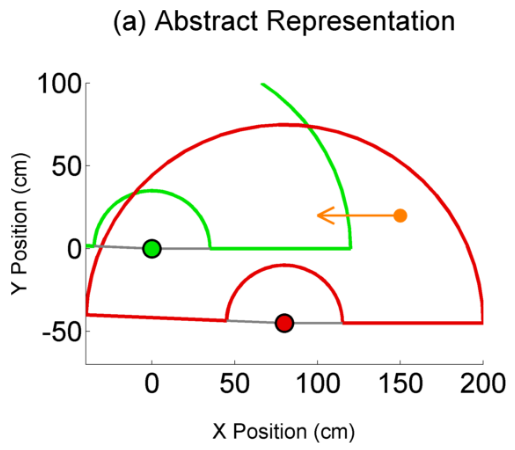
Identifying Known Object States with Confidence To extract high confidence correspondences, we track the state of the ball as either stationary or moving with fixed velocity. With these two simple motion models, we use a Switching Kalman Filter (SKF) Murphy [1998], a Markov Model composed of several Kalman Filter models, to differentiate between these two ball states. The SKF maintains the likelihood of each Kalman Filter model so we can wait until there is high confidence regarding the state of the ball and record the simultaneous observations from each robot as a correspondence.

Computing the Relative Localization Provided the matched correspondence points, relative localization is computed with the shape matching algorithm of Procrustes Analysis Ross [2004] because it is fast to compute and correspondences are already matched. The resulting translation centers the rotated points of one robot around the other robot's correspondence points. The two robots with their relative transform can now freely use each other's sensor data as long as they remain stationary.

Evaluating Benefit of Teammate Views

We empirically show the benefit of using teammate views by evaluating the task of altering the path of a ball moving towards a robot using its feet. We describe the evaluation task presented to the robot, show how teammate observations are beneficial with our computed relative localization, and highlight the importance of considering the condition of wireless communication.

Evaluation Scenario To evaluate our computation of relative localization, a ball is rolled down a ramp towards the intercepting robot as depicted in Figure 2.4(a) with an abstract representation in the robot's own egocentric coordinates and in Figure 2.4(b) with an actual overhead view. In the robot's egocentric coordinates, the robot itself is located at (0, 0) and the ball is originally observed on the right of the robot at (190, 0). The ramp is raised to three different heights to present the robot with different ball speeds: Slow, Medium, and Fast. The robot must estimate ball trajectory, project an interception point directly in front or in back, and then walk to this point in time to influence the motion of the ball with its feet. The Nao's have an approximately 180° forward facing horizontal field of view as depicted within the two semicircles, where objects too close or far are difficult to capture. This is a difficult task for the robot because the ball approaches almost directly from its right, which is close to the limit of its perceivable range. The teammate robot will be placed with an approximate relative position of (80, -45) and angle of 0° with an excellent view of the entire path of the ball.



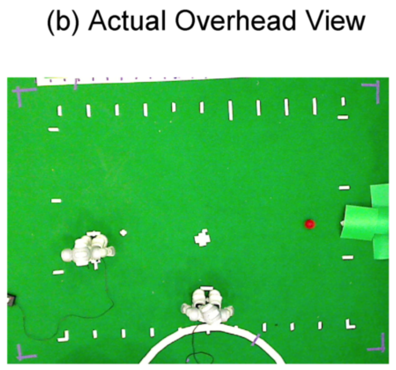


Figure 2.4: Evaluation scenario depicted abstractly (left) and actual overhead view (right).

Using Teammate Views Figure 2.5 depicts ball trajectory estimates of an actual instance of our evaluation scenario relying on each robot's individual sensor data in (a) and (b) with the actual ground truth ball trajectory in blue. The robot on its own in Figure 2.5(a) reveals that its raw sensor data in black is noisy when far away and fails to perceive the ball when close due to its cameras being occluded by its own shoulder. To project when and where to alter the path of the ball, the estimated ball trajectory by the robot on its own in green is short lived and provides only a few frames for the robot to capture an accurate ball trajectory and then make a good decision. This places an enormous burden on the SKF to capture accurate trajectory estimates from little evidence; however, it can be aided by a teammate robot with a much better view.

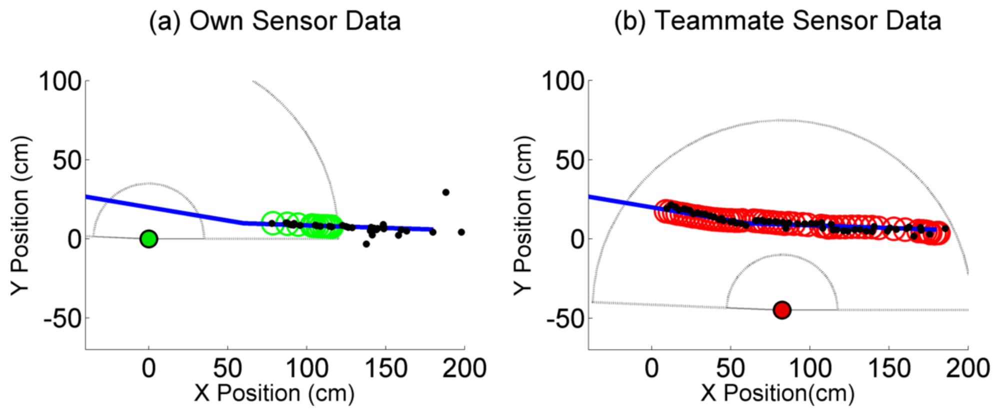


Figure 2.5: Ball observations for both robots.

Our algorithm computes the teammate robot to be at a relative location of (82.3654, -44.9545) and angle of 2.89° during the calibration step. The transformed teammate sensor data is shown in Figure 2.5(b), where it is evident that the teammate captures many more raw ball observations during the entire path of the ball. In fact, unapparent from the intercepting robot's observations is a slight unevenness in the floor slightly influencing the ball's trajectory more in the y-direction as reflected in the ground truth path. The teammate captures a much more complete and accurate set of trajectory estimates as shown in red.

Suppose teammate sensor data had been transformed using our desired relative localization of (80, -45) and 0° angle. We can see in Figure 2.6(d) and (e) that the intercepting robot would then have an estimate of the state of the ball that poorly follows from the ground truth path.

Clearly, teammate sensor data transformed with our computed relative localization helps the intercepting robot to better capture the state of its surrounding environment for making better informed decisions as shown in Figure 2.7.

With a relative transform, the intercepting robot has the benefit of using both its own and its teammate's sensor data. Listing 2.1 shows the decision process made by the intercepting robot. Ball trajectory is estimated with a SKF providing ball position and velocity estimates.

The intercepting robot is presented with three different ball speeds to see if there are any differences in success rate as faster moving balls demand even more accurate trajectory estimates with less sensor data to work with. Each ball speed type using a particular set of sensor data is

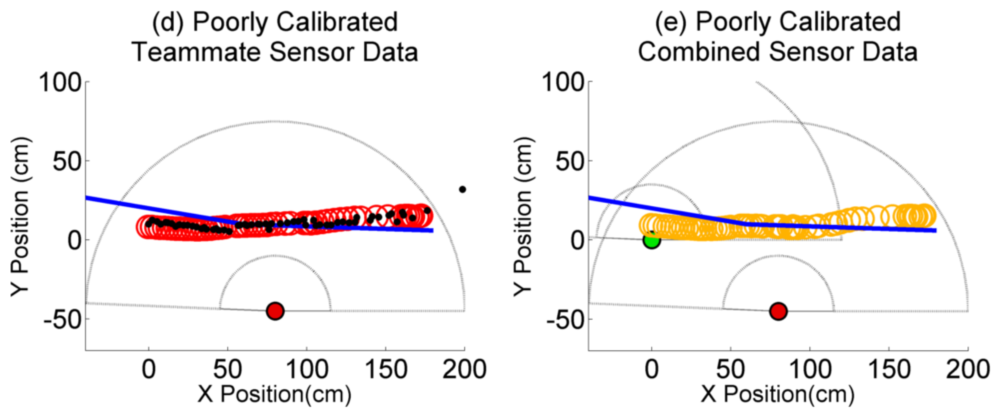


Figure 2.6: Poor fusion of ball observations resulting in erroneous trajectory estimates.

```
function AlterBallPath(Obs1, Obs2)
    BestObs = RobotClosestObs(Obs1, Obs2);
    if (BestObs == Obs2)
        BestObs = BallTransform(Obs2);

    BallState, BallPos, BallVel = SKF(BestObs);

    if BallState == MOVING {
        BallIntcptTime, BallIntcptDist =
            ProjectIntercept(BallPos, BallVel);
        RobotIntcptTime = ComputeTimeToWalk(BallIntcptDist);
        DiffIntcpTime = RobotIntcptTime − BallIntcptTime;

        if LittleTimeToReact(DiffIntcpTime)
            Walk(BallIntcpDist);
    }
```

Listing 2.1: Intercepting robot uses observations from robot currently closest to the ball, projects the estimated trajectory of ball, then decides when and where to walk to in order to alter the path of the ball.

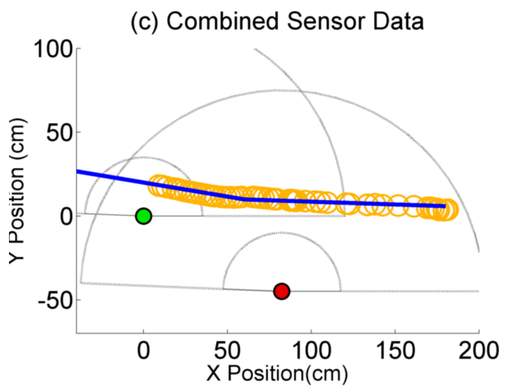


Figure 2.7: Combined ball observations resulting in accurate trajectory estimation.

evaluated 25 times each for a total of 225 instances of the task performed. While we attempt to have identical execution, factors like uneven floors results in the ball crossing robot within a range of ±20 cm requires that the robot correctly captures the state of the ball. We have a very strict criteria for an instance of the task to be considered a success. Figure 2.8 is an instance of the robot successfully altering the path of the ball and demonstrating a good projected ball interception point. In this example, the ball was moving at medium speed and effectively utilize teammate sensor data by incorporate computational and communication delays. The initial decision to actuate occurs at 1.0 s. The robot's right foot kicks ball immediately before 2.0 s. Finally, the robot's body settles at 3.0 s indicating good projection of ball trajectory.

Figure 2.9 shows an example of ball moving at a slow speed also using teammate sensor data. It is considered a failure because the robot's stopping location shows poor projected ball interception point. This suggests that when the robot made the interception decision, it did not have an accurate estimate of the state of the ball. The robot actuates just after 1.0 s and ball just happens to luckily hit robot's foot at 2.0 s. The robot eventually settles at 3.5 s, which is very far from the point where it should have intercepted the ball.

In Figure 2.10, it is evident that the success rate using only the intercepting robot's sensor data in green is dominated by nearly 10% when also taking advantage of teammate sensor data in red. For the slow moving ball, teammate sensor data has an impressive success rate of nearly 60%. When the time delay of processing teammate sensor data is also considered, the success rate dominates the intercepting robot using only its own sensors by nearly 30%. These trends suggest that teammate sensor data, especially when used effectively, helps a robot to better capture the state of the environment leading to better decisions and ultimately allow the robot to more successfully perform difficult tasks.

Accounting for the Time Delay of Teammate Sensor Data When the intercepting robot receives teammate sensor data, it was taken from a different time than the robot's own current observations. A number of factors contribute to this delay like network communication, processing, etc. The wireless network over a single router introduces almost negligible overheads of 3-5

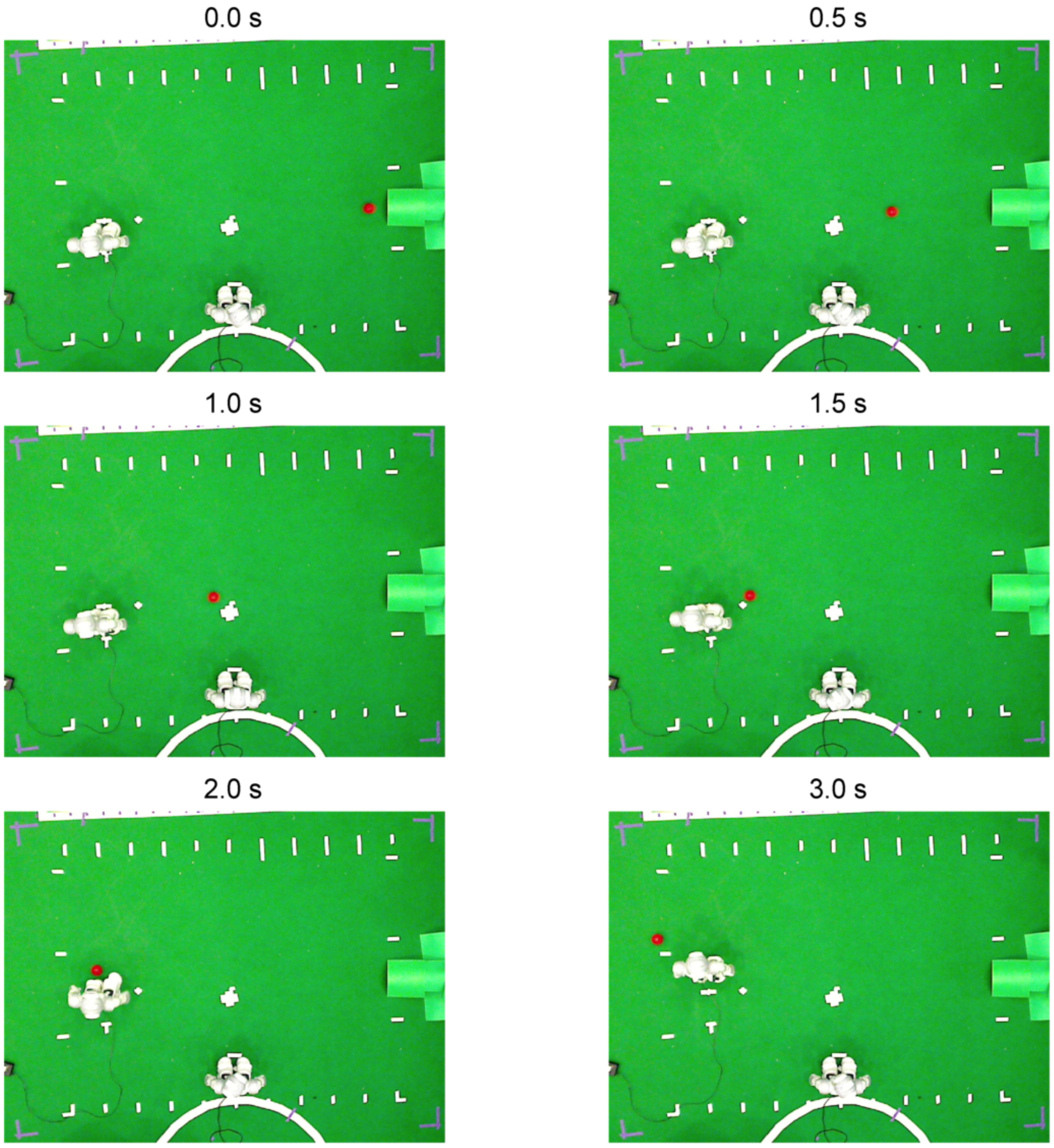


Figure 2.8: Robot successfully altering path of ball using teammate sensor data.

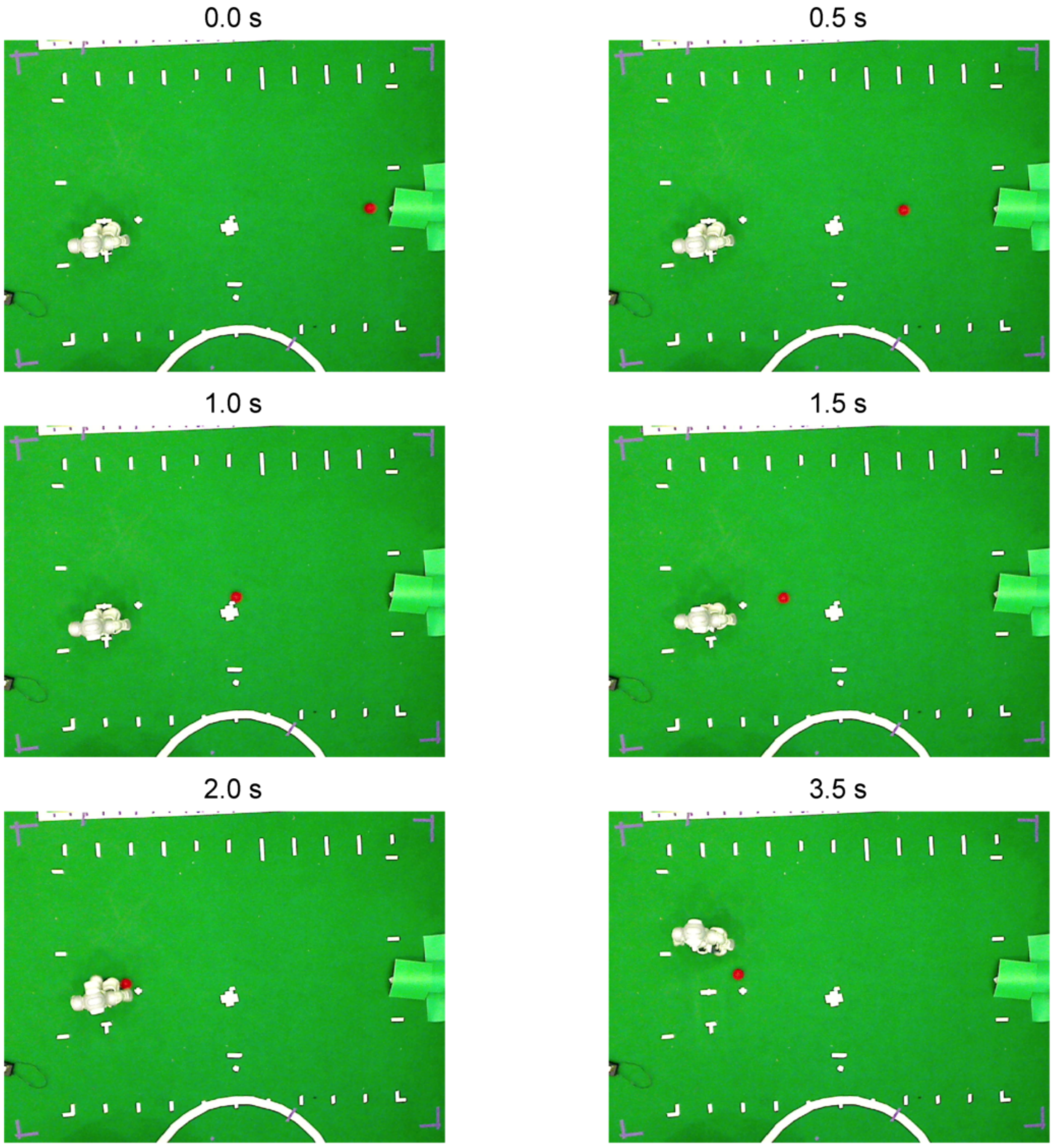


Figure 2.9: Failure because robot show poor projection of the ball interception point.

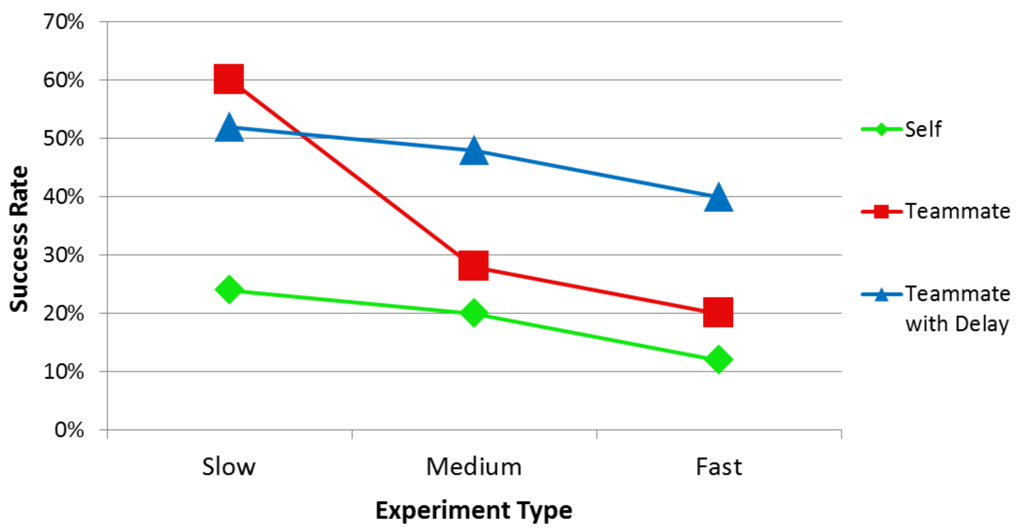


Figure 2.10: Success rate under different ball speeds.

ms. The stunning difference in success rates by considering delays in Figure 2.10 is the result of considering the total delay by the time teammate observations are actually processed by the intercepting robot. Our estimates of processing delay from our robot's control loop in addition to network communication delay of approximately 100 ms. This is a reflection of the limited compute of the Nao humanoids but does not indicate that the robot cannot work with this limitation. By recognizing the delay in processing of teammate sensor data, the intercepting robot can overcome this limitation by either delaying processing of its own observations or projecting past teammate sensor data forward in time by this delay. We choose to project teammate sensor data forward using trajectory estimates by our SKF. The success rate in Figure 2.10 when considering delay, especially with fast ball speeds, suggests that processing delay is incredibly important for successfully performing difficult tasks by effectively taking advantage of teammates with better views.

2.6 Summary

In this chapter, we introduce background on different types of mobile platforms that could be use for data collection. We explain why this leads to differences in localization abilities, which is essential for accurately labeling sensor data. We also introduce different types of sensor hardware and key factors to consider. Finally, we present a case study showing that raw sensor measurements need to be carefully processed in order to be used effectively. This motivates why the remainder of this thesis covers all aspect of collecting, analyzing, and using sensor measurements captured by mobile platforms.

Chapter 3

Data Collection with Autonomous Robots

Autonomous robots can be valuable data collectors. Their ability to navigate autonomously and continuously localize allows them to overcome the most tedious aspect of human data collection efforts: collecting sensor measurements across many locations and accurately estimating the location of each measurement. By equipping the robot with a few sensors, we should be able to frequently deploy the robot and capture sensor measurements about our surroundings. Unlike existing data collection approaches, an autonomous robot is uniquely equipped to capture both high granularity and high accuracy measurements. In order to effectively use autonomous robots for this purpose, we need to consider the unique challenges that arise from using a mobile data collection device. These challenges include understanding the accuracy of sensor measurements while moving, managing availability of sensors across long-term deployments, and ensuring data collection efforts do not negatively impact the robot's normal operation.

In this chapter, we first define how robots fall within the overall data collection design space. We then introduce a flexible data collection architecture supporting a variable number of sensing devices and analyze sensor measurements under a different static and mobile configurations. We follow by discussing how to recover accurate location estimates for each sensor measurement captured from different locations. We then show that although we collect a large number of raw sensor measurements, we will need additional processing in order to fully extract insights about our environments. Work in this chapter serves as the foundation upon which subsequent chapters further investigate how to reveal spatio-temporal insights (Chapter 4), generate robot navigation strategies based on measurements needs (Chapter 5), use other more limited mobile devices for data collection (Chapter 6), and enable time-critical decision making via fine-grain sensor maps (Chapter 7).

3.1 Data Collection Design Choices

With the recent emergence of robots due to recent developments in robot technology, it is an exciting opportunity to use robots for collecting sensor measurements. We want to show why use of autonomous robots as data collectors provide a unique combination of advantages over existing approaches. To begin, we first define the set of data collection efforts and then the set of key measurement trade-offs.

We consider the following data collection methodologies:

- **Manual -** humans tediously carry sensing devices around the environment and stop periodically to mark their location on a given map Bahl and Padmanabhan [2000], Youssef and Agrawala [2005] (also called a site survey in the wireless domain)
- **Dense Sensor Deployments -** static sensors mounted at numerous locations
- **Crowd-sourcing -** exploit cell phones that humans already carry with them
- **Robotic Testbeds -** robotic systems that can move sensors across our environments (autonomous robots are a specific type of robot that operate without human assistance)

Key measurement trade-offs include the following:

- **Density -** distance between groups of nearby measurements
- **Timeliness -** time between successive measurements at the same location
- **Cost -** combination of hardware requirements, installation costs, and operational costs
- **Effort -** amount of additional human assistance required
- **Accuracy -** combination of localization, procedural, and repeatability
- **Adjustability -** ease of changing the locations where subsequent sensor data is captured

3.1.1 Measurement Trade-offs

We define how each of the data collection approaches performs according to each of the key measurement properties. For each measurement factor, we state what is desirable from a data collection perspective and then where different collection methods fall within the possible design space.

Density and Coverage Measurement density refers to the distance between nearby measurements. We define low density to be measurements 5-10 meters apart while high density refers to a few centimeters. Naturally, high density is desirable as it shows detailed variations across our environments.

We consider manual data collection and static sensor deployments to have low measurement density because they scale poorly as the number of unique measurement locations increases. Manual efforts have low density since the amount of human effort increases exponentially with measurement density. Based on our empirical observations of our own wireless network administrators performing site surveys, measurement samples several meters apart is common for manual data collection. It seems surprising that we define dense static deployments to have low density but keep in mind that these terms are relative. After all, dense deployments are more dense than traditional static sensors that might be placed tens of meters apart (e.g. many building have small number of climate sensors per floor). Our definition of high measurement density as mere centimeters apart is even more fine-grain than large numbers of static sensors can pragmatically achieve.

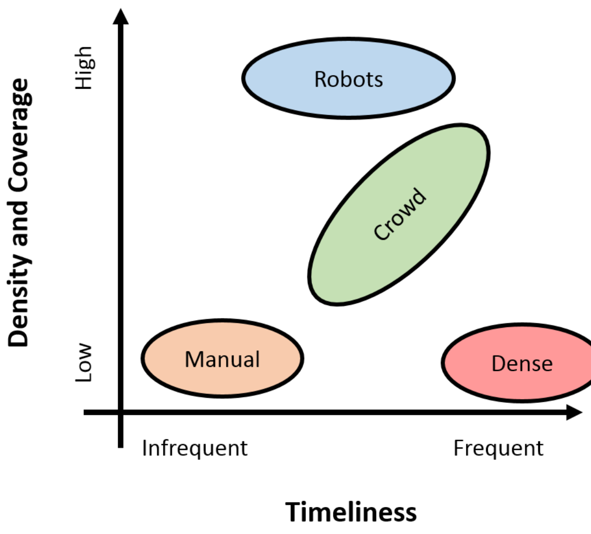


Figure 3.1: Trading off measurement coverage versus timeliness

Crowd-sourcing and robotic testbeds are considered high density as these devices can capture sensor measurements while moving across numerous unique locations. By passively capturing sensor measurements as humans move, crowd-sourcing efforts can easily achieve high measurement density. Robotic testbeds can be actively moved across numerous locations in order to achieve the same result. It is important to note that many of these robot testbeds operate in constrained spaces due to the reliance on either overhead cameras to localize Fish et al. [2006], lines marked of the ground Rensfelt et al. [2011], or humans assistance to move the robots between measurement regions Sen et al. [2012]. With the recent development of autonomous robots that navigate with little setup required, robot data collection will soon be capable to coverage even larger areas of our buildings.

Timeliness Timeliness refers to the time between consecutive measurements at the same location. High frequency measurement updates at the same location are desirable as we can better see exactly how our environments change over time.

The most infrequent approach is manual data collection because it usually takes so much time and effort to collect sensor measurements. In fact, manual efforts are often one-time occurrences repeated sporadically only when needed (e.g. investigating possible wireless dead spots when faced with connectivity issues). Moderately frequent measurement update efforts include robotic testbeds where timeliness is limited by limited deployment opportunities due to finite battery life but increasing the number of robots can potentially help to increase measurement timeliness. Crowd-sourcing is a passive operation where timeliness cannot be guaranteed but scales quickly as the number of users increases. Finally, static sensors excel at ensuring timely measurements since they always remain at the same location and therefore continuously collect sensor data.

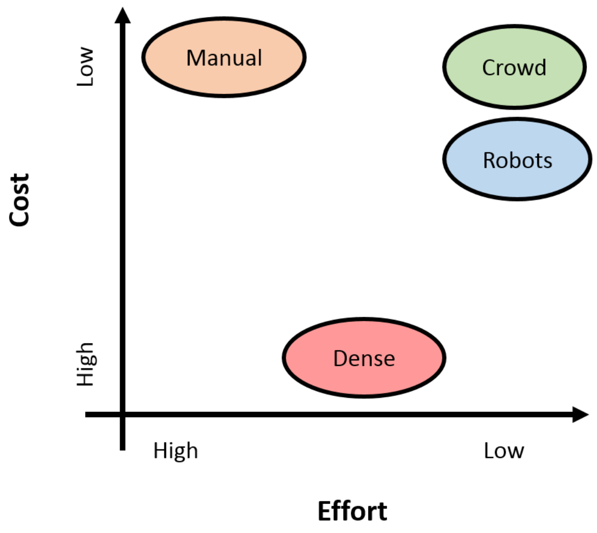


Figure 3.2: Trading off measurement cost versus human effort

Cost There are both installation and operational costs across both hardware and software. Intuitively, lower cost efforts are desired and helps make it easier to deploy across realistic environments.

Low cost approaches include manual efforts and crowd-sourcing. Manual efforts require the least amount of hardware costs (as a single sensor is manually moved to different locations) that might be offset by human labor costs if performed periodically. Crowd-sourcing opportunistically collects measurements from devices that humans already have, which effectively makes these efforts mostly free.

Robots are a moderately expensive approach. While there is little cost in operating the robot, there is a steep initial cost for the powerful sensors, computing hardware, and structure. We are hoping that emergence of robots for other purposes will similarly make data collection with robots effectively free (just like it is for crowd-sourcing today).

Dense static deployments have high hardware costs (one sensor for each location) and high maintenance costs (either initial wiring for installation or periodic batter replacements).

Effort Humans do not like performing tedious tasks, labor is expensive, and introduction of humans errors is always a potential threat so fewer humans in the loop is better. The only approach requiring significant human effort is manual data collection. It could be argued that deployments of dense static sensors requires some human assistance for installation.

Accuracy Accuracy covers several factors: localization, procedure, and repeatability. For localization, we define 2 to 3 meters as low accuracy and 10 cm as high accuracy. Procedural accuracy addresses the possibility of introducing human or environmental errors (e.g. humans accidentally wrapping hands around sensor during collection). Repeatability refers to capturing measurements at the same location in the same way each time (e.g. same sensor orientation and position). Accuracy along all of these factors provides confidence and reliability in the sensor measurements captured.

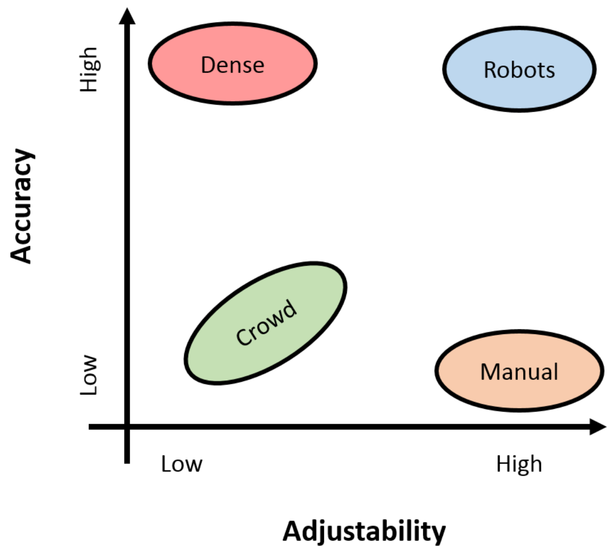


Figure 3.3: Trading off measurement accuracy versus adjustability

Manual and crowd-sourcing data collection are low accuracy for several reasons. Both suffer from poor localization although cell phones may be able to increase their localization accuracy as technology improves Youssef and Agrawala [2005], Wang et al. [2012], Rai et al. [2012]. Nevertheless, they both suffer from possible introduction of procedural errors introduces by human influence of sensor measurements. In one of our experiments, human participants inadvertently wrapped their hands around the climate sensor, which resulted in erroneous measurements. Both efforts also exhibit poor repeatability as humans are not well-suited to ensure consistent measurement conditions over time. These types of inaccuracies are inevitable for these two approaches because the devices interact so closely with humans.

In contrast, dense deployments and robotic testbeds do not require the direct involvement of humans. In fact, static sensors and robots can afford to have bulky form factors, which provides the opportunity to create structural designs that can actively prevent humans from accidentally influencing the captured sensor measurements. Localization accuracy can be achieved for both. Static sensors do not move so we can perform highly accurate one-time localization for each device. Robots continuously localize as it is essential for their ability to navigate autonomously.

Adjustability Adjusting the measurement locations over time is desirable because measurement needs may change over time. If a particular location suddenly experiences abnormal variations, it would be helpful to capture additional measurements around that location. Low flexibility means that it requires significant cost or effort to change measurement locations. High adjustability is desirable as it allows us to seamlessly change data collection strategies over time.

Dense deployments and crowd-sourcing offer little adjustability. Static sensors are fixed to a location while one has little influence over where crowd-sourced users end up moving.

Manual efforts and robots provide high adjustability. Humans can effortlessly changes where subsequent measurements are captured from. Robot testbeds can easily adjust where to collect measurements with a little bit of programming.

3.1.2 Advantages of Autonomous Robots

Now that we have defined the properties of each approach, we see how different approaches occupy different segments of the overall data collection design space. While no single approach provides all of the desired properties, autonomous robots provide a powerful mix of desirable properties that include high measurement density, high accuracy, high adjustability, low cost, and low effort. The one limitation is low timeliness because a single robot cannot simultaneously capture measurements from multiple locations. With these advantages, it is clear that use of robots for data collection presents a pragmatic solution to help capture sensor measurements from our surrounding environments.

3.2 Collecting Sensor Measurements

Given that autonomous robots can be valuable data collectors, we need to create a system that can harness these advantages. Our goal is to develop a long-term data collection system that provides fine-grain insights about our environments. To the best of our knowledge, few other efforts target such extensive coverage across large enterprise environments while capturing sensor data with centimeter-level accuracy by using a mobile platform that moves. Robot mobility is not only a key enabler for achieving these goals but also introduces a unique set of challenges for us to address. In this section, we introduce a data collection framework that supports passive measurement capture from a variety of sensing devices while also controlling where the robot collects measurements from. We also highlight the importance of understanding sensor specific configurations and parameters as well as the reliability of sensor values captured while moving.

Data Collection as a Side Effect

The design of our data collection framework is influenced by the fact that the robot can passively gather sensor measurements as it performs its various other tasks. Our goal is to seamlessly operate in the background without affecting the robot's normal operation. This would make data collection with robots effectively free if they already exist in our environments. We can build a non-intrusive data collection systems because it primarily depends on recording relevant robot localization data and corresponding sensor measurements. Only when we wish for the robot to pro-actively move around based on sensor measurement needs does the data collection system publish navigation commands that affect the robot's operation. Our system is designed to be a mostly standalone system with minimal interfaces.

3.2.1 System Design

Our data collection system is divided into two core components: data acquisition and data analysis. As depicted in Figure 3.4, data acquisition is the execution piece that continuously records where the robot is and any available sensor measurements. Data analysis is an offline process that extracts and updates insights about our environment that may also influence future robot navigation based on measurement needs. Having a separate online and offline process provides several advantages. First, we ensure that real-time data collection remains computationally light to minimize the effect on higher priority processes like robot vision. Second, offline data analysis allows us to defer how exactly we process the sensor measurements, which provides additional flexibility. Finally, a simple feedback loop creates the opportunity for results of data analysis to influence where the robot goes to collect subsequent sensor measurements.

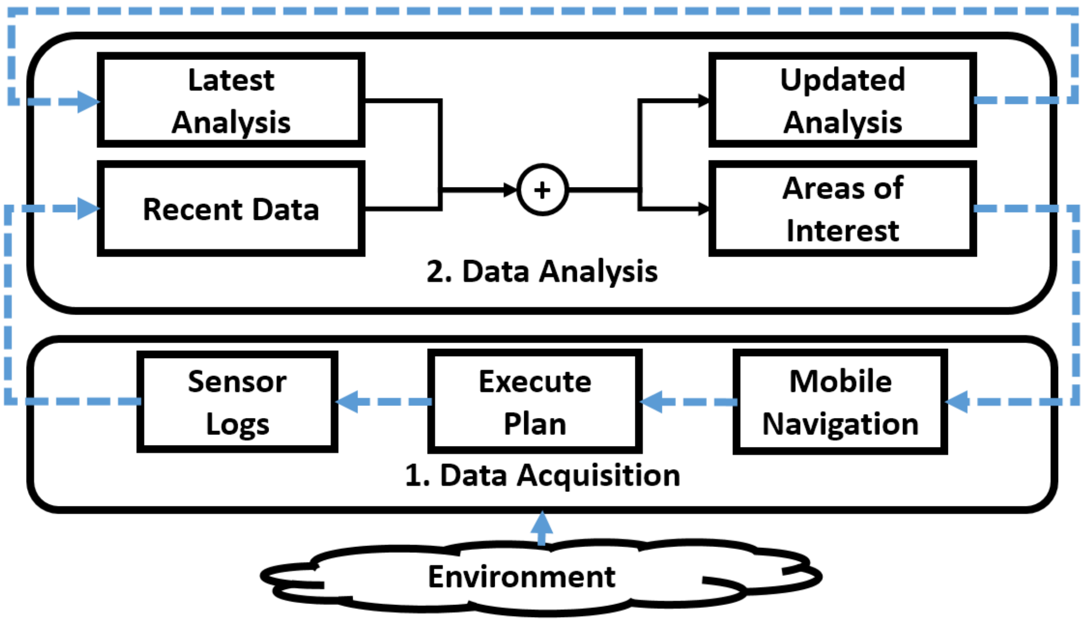


Figure 3.4: Data flow diagram of our mobile data gathering system.

Data acquisition records sensor readings so that data analysis can recover an accurate view of the environment offline. The goal of this module is to be computationally lightweight, record sensor measurements in a consistent format across long-term deployments, and properly control sensor specific parameters. Achieving these goals requires careful attention to the sensor hardware being used. For example, certain sensors have unique measurement behaviors that may take time to reach convergence (e.g. climate sensors). Others may have multiple sensor parameters to consider (e.g. dozens of wireless channels). Optional goals include directing the robot's navigation based on measurement needs (e.g. prioritize less recently visited locations).

Data analysis uses the data collected by the robot to build a model of the sensor observations of the monitored area. This is the focus of Chapter 4, where we try to make sense of fine-grain sensor measurements that vary over both time and space. Keeping data analysis as a separate offline module allows us more time to evaluate a myriad of spatio-temporal techniques while the robot is already in the process of long-term data collection.

3.2.2 Sensor Specific Measurement Records

A robot collects sensor measurements under different conditions (e.g. stationary or moving). Understanding how robot operating conditions influences these sensor values establishes confidence in the conclusions reached from insights that are extracted from the measurements collected. In this section, we discuss the wireless monitors and indoor climate sensors that are used in this thesis. Through these very different sensors, we encounter a number of challenges that we believe is representative of a wide range of other sensors.

Figure 3.5 provides an overview to show how logs are recorded by combining sensor measurements and robot context that are updated at different frequencies. Each stream of sensor measurements is annotated with a corresponding timestamp so that we can merge these data streams offline. With our design, we can easily add different sensor hardware where all we need is an agreed upon sensor message type.

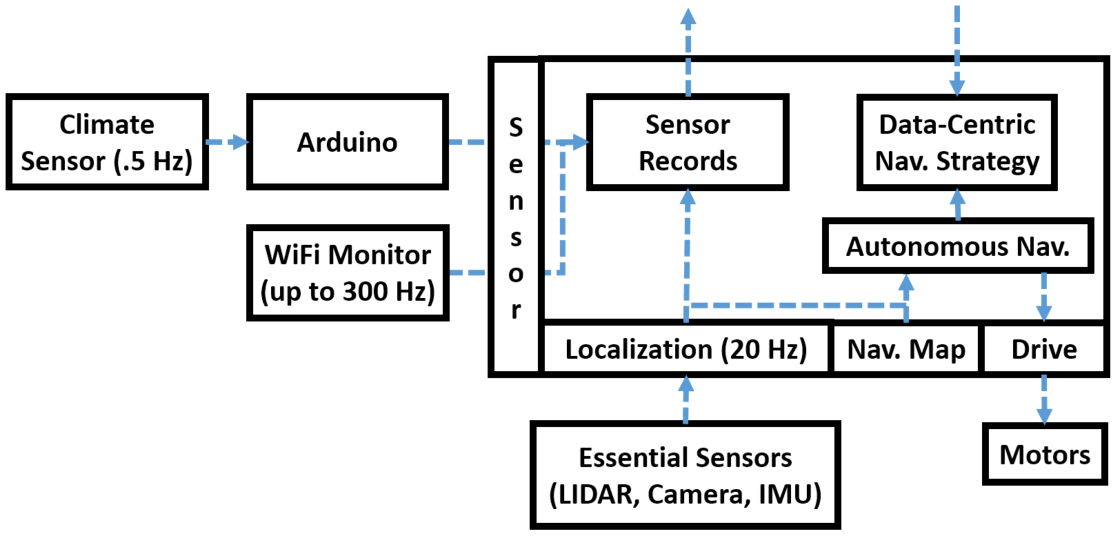


Figure 3.5: A common sensor message interface supports log records from a variety of sensors.

3.2.3 Wireless Monitors

Wireless measurements are characterized by extremely fast update rates, where robot motion introduces negligible effects when moving at speeds of up to 1 m/s. Collecting wireless measurements requires that we consider what data to record, how to configure the device, and how robot motion affects wireless measurements. Contending with the large number of wireless measurements and controlling wireless channel switches are unique challenges when collecting wireless measurements.

Wireless Records

The number of wireless samples depends on wireless usage by surrounding devices. In our deployments, we were capturing wireless samples at rates of over 300 Hz, which required careful attention to what fields to record in an effort to preserve disk space.

Each wireless measurement collected is accompanied with numerous fields. The set of fields that we record include channel frequency, data bit rate, received signal strength indicator (RSSI), source MAC address, destination MAC address, and base station service identifier (BSSID). These fields were chosen for several reasons. Addresses are important because RSSI depends on the distance between the transmitting and receiving device (where the receiving device is our wireless monitor). We found it important to record addresses as integers instead of strings to reduce the number of bytes for each measurement record. Bit rate and channel information are fields relevant for understanding wireless usage by surrounding devices.

Additional disk saving arose by filtering out wireless measurements irrelevant for our purposes. Recall that a wireless monitor passively captures bi-directional packets. In general, the wireless infrastructure APs serve as the backbone for providing wireless connectivity. Our thesis work focuses on the wireless infrastructure so we only record packets transmitted from wireless access points. To filter just these packets, we check that BSSID and source address fields match and that the SSID matches those of our target wireless network. If we suspected that interference from other users was the source of our connectivity issues, we would have applied a different filter targeted at transmissions from wireless users instead of APs. Of course, wireless measurements from users would be more difficult to analyze because users move around and tend to be more transient. In contrast, APs are generally stationary so long-term data collection efforts can accumulate a more consistent set of measurements over time.

Channel Switching

A wireless device can only be set to one of many possible wireless channels at a time so channel switching is needed to fully capture the wireless environment. Switching between wireless channels is one of the parameters that we need to control for effective data collection. The basic idea is to quickly iterate through the set of wireless channels.

Unfortunately, channel switching is not instantaneous so the device needs to spend enough time on each channel to capture a sufficient number of wireless samples but also switch fast enough so that the robot does not move a significant distance between successive samples on the same channel. With 35 channels and a robot limited to speeds of .75 m/s, channel switching at a fixed rate of 30 Hz will only take 1.2 seconds to cycle through all channels in which time the robot will move less than 1 meter. We found such a strategy sufficient for capturing dense spatial coverage across all channels after multiple traversals.

Wireless Infrastructure Configurations

The wireless infrastructure is a collection of access points that are directly connected to the wired backbone that provides Internet connectivity. Even though APs do not move, there are parameters that affects how transmitted wireless signals propagate across the environment. There parameters include transmit power and wireless frequency. Many networks today have controllers that automatically adjust these parameters over so our surrounding wireless conditions are continuously changing. In general, it is a challenge to disambiguate actively controlled changes (e.g. AP re-configurations) from environmental variations (e.g. doors opening and closing and furniture adjustments). Tracking fine-grain variations over time may help to reveal such re-configurations over time.

Effects of Robot Motion

We need to verify that neither the robot's structure and materials nor its mobility affects the accuracy of the wireless measurements. Unfortunately, there is no source of ground truth wireless measurements so we find other ways to measure the reliability of the measurements. Because the robot uses the same sensor hardware mounted at same location on itself, we are most concerned about the relative accuracy of the measurements. This includes understanding if the robot's orientation introduces any abnormal effects as well as relative calibration across different sensor hardware options. With this knowledge, we will be able to recover a consistent set of measurement values even if collected from different sensor hardware mounted on different robots.

Effect of Orientation on Wireless RSSI We place the robot in various orientation to show that there are no abnormal effects by either the robot's structure or various materials. As long as any effects introduced by the robot consistently affect the measured signal strength, we can correct these later on through calibration. Previous efforts showed that human bodies have little impact on RSSI because while it is true that humans affect the direct line of sight component of wireless signals, these are largely mitigated by multipath effects Sen et al. [2012]. As a result, we do not expect that robot orientation introduces significant effects on wireless RSSI measurements.

Figure 3.6 shows how RSSI for three different APs at three different locations vary as the robot spins in place. RSSI is shown in polar coordinates where distance from the origin is RSSI (lower is better) and angle reflects the global orientation of the robot. Notice there is little variation as the robot spins, regardless of whether or not the robot is near or far from the AP. These observations indicate that wireless measurements captured at a single location are consistent regardless of our robot's orientation. Therefore, the remainder of this work does not incorporate the dimension of robot orientation during subsequent measurement analysis.

Comparing Sensor Hardware Different sensor hardware under identical conditions may return different sensor values. Comparing these differences while the robot moves provides several valuable insights. This includes understanding systemic differences due to capabilities, relative

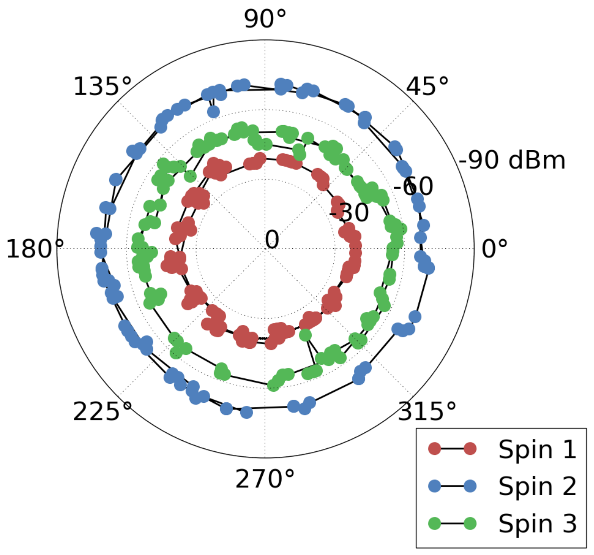


Figure 3.6: RSSI as the robot spins in place for 3 different locations and APs

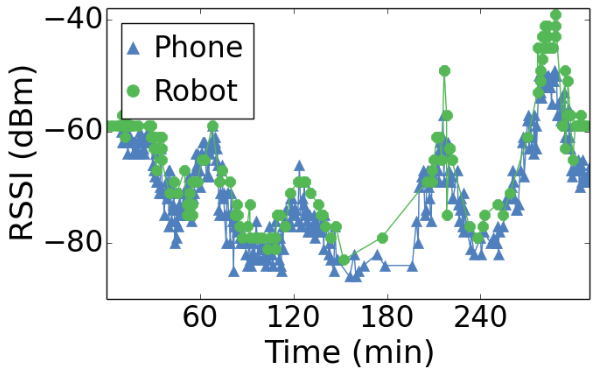
calibration like scaling or offsets between sensors, and any aberrations in sensor values. Understanding sensor hardware differences allow us to better understand how reliably the wireless monitor reflects surrounding wireless conditions.

We perform a simple experiment that simultaneously captures wireless measurements from two different wireless monitors mounted right next to one another on the robot. The wireless monitors using different hardware with one being a wireless-N dongle equipped with a 3x3 MIMO antennae while the other is a Nexus S cell phone equipped with a single antennae.

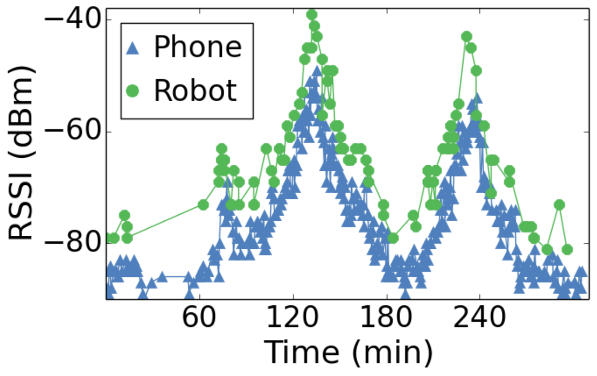
Figure 3.7a and 3.7b compares the RSSI for two APs captured as the robot moved across the environment that are typical of most outcomes that we see. Overall, measurements from both devices correspond well and follow the same general shape suggesting general agreement about surrounding wireless conditions. In addition, the robot seems to collect slightly fewer wireless samples than the phone.

Upon closer inspection, we do see some clear differences that cannot be completely attributed to simple calibration issues. For example, while RSSI matches well in Figure 3.7a, there is a RSSI offset difference in Figure 3.7b. The goodness of RSSI correspondences seems to depend on the AP being captured. These different behaviors likely arise from the more advanced wireless-N dongle that is taking advantage of its 3x3 MIMO antennae to better aggregate multipath effects. As a result, one cannot simply mix and match measurements from different wireless monitors because technological differences may result in different values.

By comparing simultaneous measurements, we could compare the sensitivity and precision of different sensors, understand differences in the sampling rate from passively collected signals, and show that both these devices effectively capture the surrounding wireless environment although additional efforts is require to merge measurements from different hardware.



(a) AP #1 - Matching RSSI



(b) AP #2 - Higher Robot RSSI

Figure 3.7: Simultaneous RSSI comparing WiFi dongle and a cell phone for two APs.

3.2.4 Indoor Climate Sensors

Climate sensors measure temperature and humidity of the surrounding environment. These measurements update at a slow rate and take time for the readings to reach equilibrium. This requires careful consideration about how the robot moves around for collecting such measurements. Unlike wireless monitors, there are no parameters to control (e.g. switching wireless channels). Therefore, the biggest challenge in collecting climate sensor data will be controlling the robot's navigation to ensure reliable measurements.

In our experiments, we used two different digital temperature and humidity sensors, the AM2303 and AM2315, connected to an Arduino UNO that serially publishes messages for our autonomous robot to record. Both sensors capture temperature and relative humidity values that are updated at a maximum fixed frequency (0.5 Hz in our case). We now show how long it takes indoor climate measurements to reach equilibrium and then discuss how movement affects these sensor values.

Time to Reach Equilibrium

Accurate temperature and humidity measurements require time for the measurements to reach equilibrium. This may necessitate that the robot move extremely slowly or remain completely stationary in the worst case. We perform two experiments to better understand the convergence of climate measurements. First, we expose the sensor to a heat source to see how long temperatures return to equilibrium. Next, we compare climate measurements between a stationary and mobile robot to see if sensor values might be affected from being moved around. The idea is to understand how quickly climate sensors react to changing conditions as well as how accurately they capture surrounding conditions.

Convergence After Exposure to External Heating Source By exposing the sensor to an external heat source, we can see exactly how long it takes for the sensor reach equilibrium. This will let us know in the worst case environments with extreme climate variations how long the robot would need to remain stationary to collect highly accurate climate measurements.

Results from a simple experiment exposing the robot's climate sensor to a heat lamp is shown in Figure 3.8. The dashed line indicates when the heat lamp was turned on and then solid line indicates when the heat lamp was turned off. Notice it takes about 400 seconds for the temperature sensor to reach the highest temperature. It takes 300 seconds to reach equilibrium after the heat lamp is turned off. Climate controlled environments are unlikely to have such extreme temperature variations so the robot may not need to spend so much time waiting in practice. Given that people often perceive a stopped robot as broken, our compromise is to direct the robot to move at its slowest speed of 0.25 m/s when collecting climate measurements.

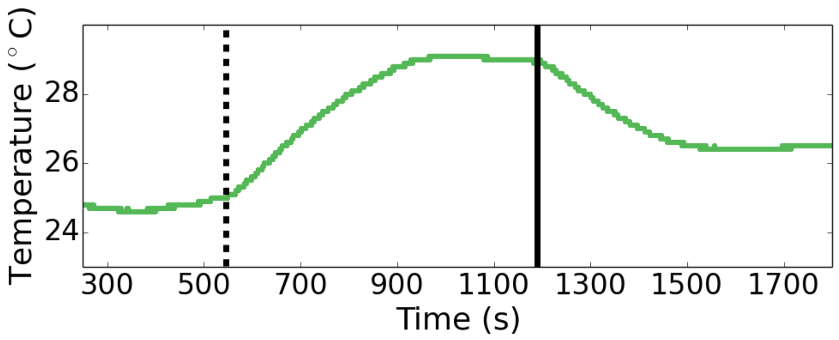
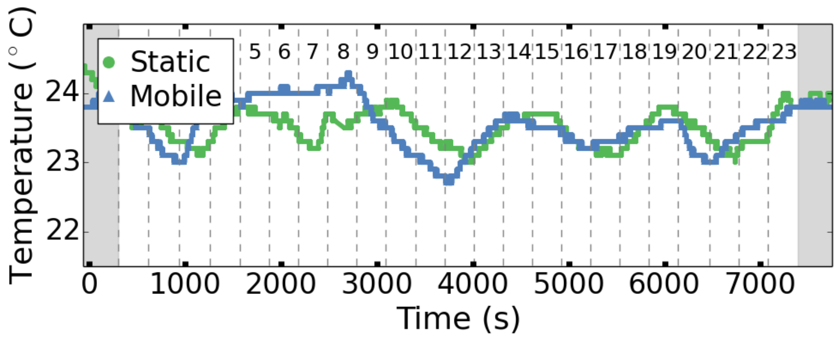


Figure 3.8: Climate sensor after heat lamp turned on (dashed) and then off (solid).

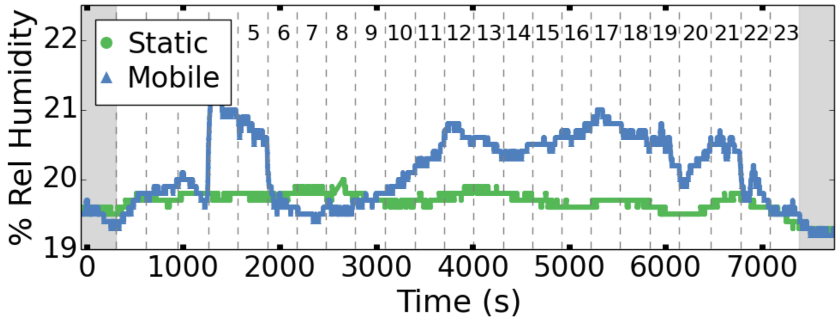
Reliability of Measurements Captured by a Mobile Device When we initially started testing the indoor climate sensors in the wild, we were surprised at the variability of temperature and humidity over time. Unfortunately, it was difficult to know if these variations arose from poor sensor hardware or that our indoor climates do in fact vary wildly across different locations. It was also unclear if these were systemic variations across the entire building or location specific climate differences. To investigate these issues, we simultaneously deploy two robots with identical climate sensors where one remains stationary while the other moves around. Both devices start and end at the same location so we will see if sensors values agree even after one of the device has been moving around. Our results show that indoor climate varies across a single location over time and that there are differences across multiple locations.

Figure 3.9 shows results of a 2 hour experiment comparing temperature and humidity measurements. The mobile robot starts and ends at the same location as the static robot (highlighted in gray) but also visits 24 unique waypoint locations (spaced roughly 1.5 meters apart). The mobile robot remains at each waypoint for 5 minutes.

Given that the sensor values match well when both robots are at the same location, we can be confident that differences at the waypoints can be attributed to differing climate conditions. There are some notable insights we can gather. First, while humidity variations are quite small for the stationary device, relative humidity differs by over 1% at different waypoints suggesting that conditions differ based on location. In contrast, temperature varies in a periodic manner by nearly $1°C$ for the stationary device. Temperatures differ by nearly than $1.5°C$ depending on the location. These results show that there are actually a lot of differences in temperature and humidity across our environments.



(a) Temperature



(b) Humidity

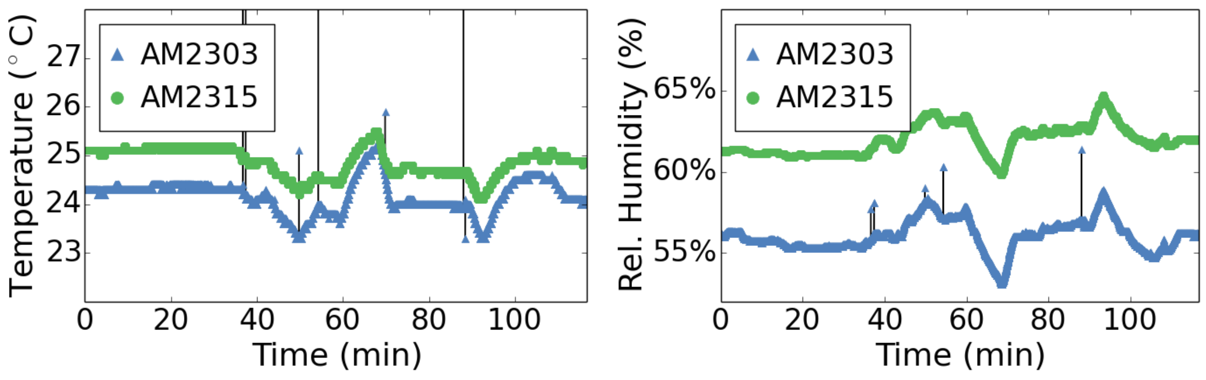
Figure 3.9: Indoor climate comparison from a stationary and mobile data collector.

Effects of Robot Motion

Similar to the wireless monitor, we need to understand how measurement values differ depending on the climate sensor being used. In Figure 3.10a, we compare two climate sensors (the AM2303 and AM2315) that measure temperature and humidity while the robot moved across the environment. Over 2 hours, the relative changes across the measurements seem to correspond fairly well and perhaps could be calibrated with simple scaling and offset alignments. Similar to the wireless monitors, both sensor values captured by both sensors reflect similar relative changes. One clear difference is the small number of erroneous measurements captured by the AM2303. As a result, the AM2315 is definitely the slightly preferred sensor due to more reliable measurements.

3.3 Processing Sensor Measurement Records

Our data collection framework records separate streams of sensor data and robot localization at different rates. In order to match each sensor measurement with their corresponding location, we



(a) Temperature (b) Humidity

Figure 3.10: Simultaneous temperature and humidity from AM2303 and AM2315 sensors.

merge this information by using timestamps based on the robot's internal clock so that all data streams have a common reference. We first discuss the theoretical accuracy of location estimates for each sensor measurement and then present the matching algorithm used in this thesis.

3.3.1 Measurement Location Accuracy

The accuracy of measurement capture location is dictated by both the robot's localization accuracy and the update rate of robot localization and sensor measurements. If we represent the update frequency of robot localization as $freq_{localization}$ and sensor updates as $freq_{sensor}$, then the number of localization samples for each sensor measurement is:

$$\frac{\#\ localization\ samples}{1\ sensor\ update} = \frac{freq_{localization}}{freq_{sensor}} \tag{3.1}$$

The location accuracy of each sensor measurement is then bounded by the maximum distance between successive sensor samples S_i, S_{i+1} and robot localization noise ϵ, which is 10 cm for our robot.

$$error_{pos} < 2 * \epsilon + L2norm(S[i]_{pos}, S[i+1]_{pos}) \tag{3.2}$$

For more accurate sensor position estimates, a ratio below 1 is desired since the location error is bounded primarily by the robot's localization accuracy. With a ratio above 1, then we have uncertainty due to multiple position estimates corresponding to one sensor measurement. For example, our robot localization updates at 20 Hz, temperature/humidity sensors at 0.5 Hz, and WiFi monitors at 100 Hz. The WiFi monitor updates much more frequently that the robot's localization so localization accuracy accounts for most of the positional errors. For the temperature/humidity sensor, 2 seconds can pass between successive samplings. Our robot is limited to speeds of up to

0.75 m/s so the total positional errors for each sensor measurement is $2 * .01 + 0.75 * 2 = 1.52$ m. To mitigate the effects of inaccurate climate measurement locations, we limit the speed of the robot when collecting climate data to 0.25 m/s to reduce location inaccuracies. In contrast, the robot's speed is not an issue that we focus on when collecting wireless measurements.

3.3.2 Matching Sensor Measurements with Corresponding Locations

All sensor streams are marked with the robot's internal timestamp that will be essential for merging with localization data. For example, we have a sequence of robot localization measurements as pairs of time and position $(time_{l,1}, pos_1)$,$(time_{l,2}, pos_2)$ ... $(time_{l,n}, pos_n)$ and a sequence of sensor measurements as a pair of time and data value(s) $(time_1, data_1)$,$(time_2, data_2)$ ... $(time_m, data_m)$. With different update rates, these two sequences can have different number of entries. Algorithm 1 shows how a single pass is sufficient for matching each sensor measurement with a corresponding location.

Algorithm 1 Merge localization and sensor measurement records by timestamp.

Require: Robot localization records R as sequence of $(time, pos)$ pairs. Sensor records S as sequence of $time, data)$ pairs.

```
 1: function MERGEBYTIMESTAMP($R, S$)
 2:     $merged \leftarrow []$
 3:     $idx_{robot} \leftarrow 0$
 4:     $idx_{sensor} \leftarrow 0$
 5:     while $idx_{robot} < R.length$ do
 6:         while $S[idx_{sensor}]_{time} > R[idx_{robot}]_{time}$ do
 7:             $merged+ = \{S[idx_{sensor}], R[idx_{robot}]\}$
 8:             $idx_{sensor}++$
 9:         end while
10:         $idx_{robot}++$
11:     end while
          return $merged$
12: end function
```

3.4 Challenges in Making Sense of Fine-Grain Measurements

We collect a large number of fine-grain sensor measurements when using robots for data collection. Unfortunately, the value of these measurements in their raw form is limited without additional analysis like those described in Chapter 4. In this section, we wish to show why this seemingly large amount of raw data is actually quite small once one starts to drill down on various factors (e.g. location, time of day, wireless access point, etc.).

Experiment Setup To support our claims, we use sensor measurements collected from actual robots during long-term deployments across large environments. We performed two lengthy measurement collection studies across two buildings: the Gates Hillman Center (GHC) in Pittsburgh, PA and the MetroTech Center (NYU) in Brooklyn, NY. As shown in Table 3.1, we collected extensive amounts of WiFi data in GHC across five weeks in which the robot was in operation for over 40 hours and moved over 70 kilometers. In NYU, the robot collected data across 3 weeks and traveled over 60 kilometers.

Floor	6th	7th	8th	9th	Total
Time (hours)	12.3	14.4	7.9	7.2	41.9
Distance (km)	22.4	27.7	14.0	12.4	76.5

Table 3.1: Duration and distance of the robot's 5 week deployment in the GHC building.

Data, Data, Everywhere But Here Despite the large aggregate number of measurements collected, the number of measurements measurements after drilling down into a particular location and sensor configuration is inevitably limited when collecting measurements with robots that move around. Repeated traversals over the same area across multiple deployments help to increase measurement density.When we consider that environments change over time and robots have limited data collection time due to battery life, measurement sparsity is an inherent limitation that can be mitigated mostly be adding more data collection devices. We now illustrate wireless measurement sparsity from our own data collection experiences across four floors of the Gates Hillman Center (GHC).

We begin by looking at aggregate statistics that show how the number of wireless samples differs across APs. As shown in Figure 3.11, some APs are measured frequently (e.g. one AP has over 1400 samples). while others APs have fewer than 50 samples. These differences arise because we cannot pick and choose which AP we collect measurements from; instead, our wireless monitor simply passively collects whatever wireless measurements happen to be transmitted. APs with many measurements are being heavily used by wireless users for connectivity. APs with few measurements are capturing the sparse number of beacon packets that APs periodically broadcast to announce their presence. As we have begun to show, the total number of wireless samples begin to look small once we begin to segment them (by AP in this case).

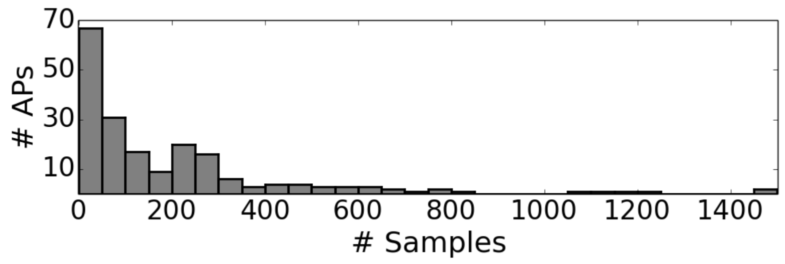


Figure 3.11: Number of wireless samples collected for each AP.

Next, we want to to see if more frequently visited locations result in more wireless samples. Figure 3.12 compares the number of visits with and without wireless samples being collected for one of the AP's with a total of 542 samples. We show the number of visits with wireless samples (green) and the total number of visits (blue, unstacked). These results show that frequently visiting a location does not guarantee measurement capture.

The key takeaway is that as we continue to further refine our analysis, the sparseness of measurements becomes more and more apparent. In fact, it may not be something we can easily controlled by simply visiting certain locations more frequently. The number of samples for certain sensors like wireless monitors are fundamentally limited due to their dependence of other devices transmitting packets.

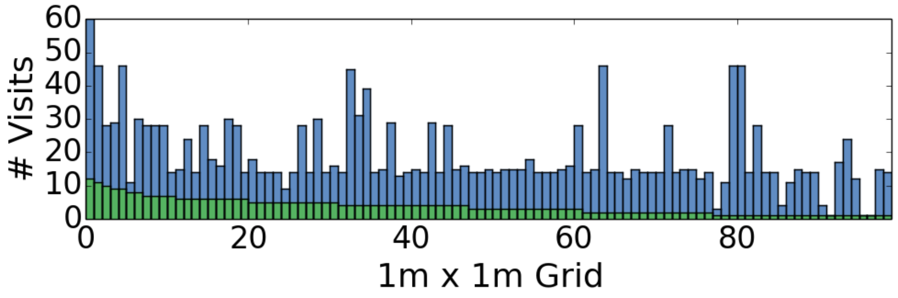


Figure 3.12: Number of visits to each location with and without wireless samples.

3.5 Next Steps

The work described in this chapter forms the foundation for the subsequent chapters of this thesis. We showed the problem of measurement sparsity once one begins to segment sensor measurements based on various factors. First, Chapter 4 tries to take advantage of the high spatial granularity of measurements collected in order to extract more meaningful spatio-temporal insights. Next, Chapter 5 prioritizes where the robot should collect additional measurements so that we more effectively utilize data collection opportunities and mitigate the measurement sparsity challenges. In addition, Chapter 6 investigates the challenges of using alternate mobile platforms like cell phones whose large numbers may help to provide additional measurement samples. These efforts will try to make better use of the measurements, more intelligently collect measurements, and consider the use of other devices. Finally, Chapter 7 presents a concrete application to show the benefits of data-driven algorithms that utilize these fine-grain sensor measurements.

3.6 Summary

In this chapter, we began by discussing the unique advantages of using autonomous robots for data collection when compared to existing techniques. We then showed how specific sensors behave when collected by a mobile platform. We presented a data collection framework that seamlessly collects sensor measurements while the robot performs its routine tasks. Based on actual data collected by robots, we notes that measurement sparsity is a fundamental challenge that we hope to mitigate in next few chapters.

Chapter 4

Representation and Processing of Fine-Grain Measurements

In this chapter, our goal is to extract insights about our environment from the fine-grain sensor measurements collected by a mobile platform. These measurements may differ by a little as a few centimeters or milliseconds or be from completely different spatio-temporal contexts. The key challenge is computing context-specific conclusions based on sensor data where no two measurements are captured at the same location or time. Data analysis must overcome the challenges of spatio-temporal measurement sparsity (as previously discussed in Chapter 3), where the number of measurements captured becomes small as we drill down on more and more specific times and locations. In this chapter, we present a discrete approach to quantify sensor measurement values over both space and time. We present a spatial representation that decomposes the environment into discrete regions specifically tailored for a mobile data collection platform. We show the fine-grain insights we can extract about our environments based on real data collected after long-term deployments of our autonomous robots.

4.1 Representation and Processing Choices

Unlike fixed static sensors, a mobile data collector collects measurements from numerous unique locations over time. Given that sensor values are affected by their measurement context (where and when they were collected), making sense of spatio-temporal correlations is very important for sensor data collected by a mobile platform. In this section, we discuss the trade-offs between different approaches for analyzing spatio-temporal data. We will see that how one decides to spatial-temporally represent the environment dictates how one can then analyzes the data in order to gain insights.

One can represent an environment discretely or continuously. With a discrete representation, one clearly divides the environment into clearly marked spatial boundaries (e.g. grids, areas, regions) and time slots (e.g. hourly, daily, weekly). Advantages of clear measurement groupings enable computationally light weight data analysis where one can apply a wide range of traditional statistical techniques. The primary disadvantage of discrete region boundaries occurs where either excessively large boundaries group less correlated measurements that lead to information loss or excessively small boundaries that suffer from insufficient data. As a result, discrete approaches depend on specifying appropriate boundaries.

In contrast, continuous representations preserve fine-grain locations and times but require more sophisticated analysis to extract insights. Geostatistical techniques like Kriging Phillips et al. [2012], Riihijarvi et al. [2008] and Gaussian Processes Ferris et al. [2007] perform interpolation over both space and time. The most significant benefit of a continuous space representation is that they are robust to small measurement variations and perform an exact interpolation that minimizes the squared error. Disadvantages include poorly handling small amount of data, estimating smoother models than reality, and suffering from uncertain knowledge. In additon, a known covariance model, which specifies the similarity of measurements within a neighborhood, is essential for accurate interpolation. Unfortunately, these covariance models are generally unknown in real environments.

We choose to use a discrete representation due to the characteristics of sensor measurements captured by a mobile platform, the repeated nature of sensor data collected by a mobile platform, and simpler usage of measurement insights.

To combat the challenge of measurement sparsity, a discrete representation has the flexibility to tune the size and shape of regions in order to ensure data analysis from sufficient measurement density. For a continuous representation, this would be embedded in the covariance model that is more difficult for humans to modify. Second, a mobile platform can only collect measurements from one location at a time. Our surrounding environment may also be simultaneously changing over time as well and these variations depend on the sensor values being captured. For example, indoor climates tend to vary periodically over the course of a day while wireless configurations changes abruptly after several weeks. With the combination of discontinuous sensor updates over time and a changing environment, it is difficult for both discrete and continuous representation to make sense of the sensor measurements. However, the discrete approach is both more computationally light than the continuous approach while also providing more manageable visualization of variations over both time and space.

Data collection with mobile platforms is intended to operate repeatedly over time. Given that it is inexpensive and easy to change where additional measurements are collected from, it is important to quickly and effectively compute where and when subsequent data collection efforts should occur. After all, if we notice sudden anomalous readings from a certain location, it would be desirable to focus data collection around that location. These goals align very well with a discrete representation where data analysis is fast to compute but might possibly introduce some information loss. We envision these types of efforts that ensure up-to-date sensor maps align well with desirable usage of this sensor data. After all, all sensor data being used will inevitably be historical (whether by an hour, a day, or several weeks). When devices make decisions based on sensor data, they must reflect existing conditions or else be effectively useless. While perhaps not as accurate as continuous analysis, a discrete approach is entirely capable of discriminating trends of space and time while also fast to compute.

In summary, a discrete representation is well-suited for sensor data collected by a mobile platform. Due to the sparseness of sensor measurements across location and time and repeated nature of such collection efforts, it is important that we quickly react to changes in the environment in order to better focus subsequent data gathering efforts as well as best ensuring that devices make decisions using up-to-date view of the environment. Analysis over a discrete space allows us to use light-weight computations that provide the opportunity to react quickly as our environments change.

4.2 Navigation Adjusted Grids (NAG) for Data Collection

Having decided to use a discrete representation, we are faced with the very important task of defining spatio-temporal boundaries. While discrete times are fairly straightforward, a key challenge will be deciding how to spatially divide the environment into small enough regions that measurements are well correlated but large enough so that each region has a sufficient number of samples. While previous efforts include spatial decompositions that include grids and voronoi cells, we introduce navigation adjusted grids (NAG) that are specifically tailored to be centered around where the robot can actually reach. We show that NAG's better distribute measurements in space. We then show insights that can be reached based on actual sensor measurements collected by an autonomous robot.

4.2.1 Spatio-Temporal Data Collection Matrix

We introduce a discrete spatio-temporal representation that we refer to as a data collection matrix. We will define the parameters of this matrix as a combination of regions and time slots. As we receive a stream of sensor measurements, they will be grouped into the appropriate cell based on their corresponding location and time. Visualized as a 2D matrix in Figure 4.1, we see that each column corresponds to a particular time slot so that all spatial entries within a particular column form a spatial snapshot (e.g. all sensor data collected between 2-4 AM).

We first define a set of time slots *TSLOTS*. Each non-overlapping time slot t_i forms the set $\{t_0, t_1, ..., t_n\} \in$ *TSLOTS* and is composed of a start and end time such that $\{start, end\} \in t_i$. Depending on how the sensor values are expected to change over time, one could use a set of time slots within a global time frame (e.g. time since epoch) or a periodic time frame (e.g. time of day, day of week). A periodic time frame provides the opportunity to more densely fill the data matrix across multiple data collection opportunities in order to mitigate the problem of measurement sparsity over space and time.

Next, we define a set of regions *REGIONS*. Each non-overlapping region r_i forms the set $\{r_0, r_1, ..., r_m\} \in$ *REGIONS*. Each collected measurement is matched with a corresponding region whose boundaries it falls within. The goal is for each region to simultaneously be small enough to ensure spatial locality but also large enough to contain a sufficient number of measurements.

With regions and time slots on each axis of a matrix, we can define data collection matrix *DMATRIX* as a set of bins $\{b_{0,0}, b_{1,0}...b_{n,m}\} \in$ *DMATRIX*. Each bin $b_{i,j}$ corresponds to a specific

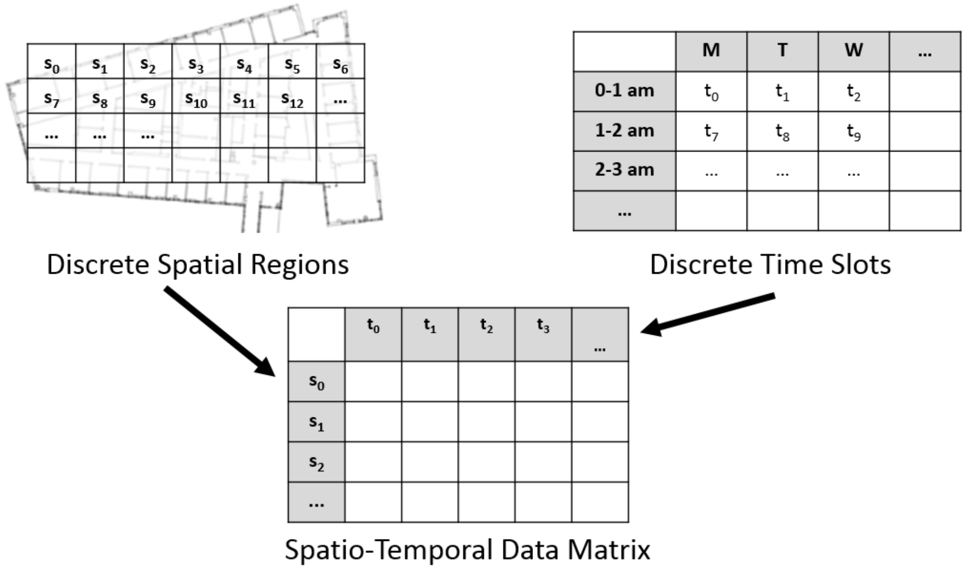


Figure 4.1: Spatio-temporal discretization of the environment.

pair of region and time slot $(r_i, t_j) \in b_{i,j}$ as visualized in 2D matrix in Figure 4.1.

4.2.2 Disaggregating a Stream of Sensor Records

We now define how a stream of sensor measurements are grouped into their corresponding cells of the data collection matrix. Define each sensor measurement s to be a tuple with time t, location (x, y), and sensor value q so that $(t, (x, y), q) \in s$. Instead of assigning each individual measurement s directly to its corresponding bin, we actually want to record a collection record w_k. Each record w_k is a start and end time indicating when the robot passed through bin $b_{i,j}$ as well as a corresponding list of sensor measurements $\{s_0, s_1...\} \in s_{list}$ so that we get $(\textit{start}, \textit{end}, s_{list}) \in w_k$. Therefore, each entry in our data matrix bin $b_{i,j}$ is a list of these collection records $\{w_0, w_1...\} \in w_{list}$. Choosing to store measurement records will be important later on when computing coverage metrics because time the mobile platform spend at a location can be independent of the number of measurements captured.

4.2.3 Generating Navigation Adjusted Grids (NAG)

We introduce navigation adjusted grids (NAG), which are specifically targeted at discretizing the space in which the robot can reach. NAGs are better suited for data collection because they align grid regions to the robot's underlying navigation graph, ensure non-overlapping coverage, divide the environment into identically sized regions, and use a simple bounding box computation for matching measurement locations with their corresponding grid region. Alternate efforts in spatial discretization are not as well-suited for data collected by a mobile platform. Many do not separate the environment into similar sizes regions (Voronoi diagrams and cellular decomposition). Others apply rigid boundaries unaware of where measurements can actually be captured, which results in poor distribution of sensor measurements (metric grids).

Definitions

Let us define the NAG, which is essentially a set of square grid cells. Define each grid cell r_i as a square region with bounding box points: top left (x_{tl}, y_{tl}) and bottom right (x_{br}, y_{br}) so the boundaries can be defined as $(x_{tl}, y_{tl}, x_{br}, y_{br}) \in r_i$. The dimensions of the square are bound by some size *SIZE* that we assume is the same across all locations *REGIONS* such that $abs(x_{tl} - x_{br}) ==$ *SIZE* and $abs(y_{tl} - y_{br}) ==$ *SIZE*. The robot's navigation graph *NAV* is the set of vertices v_i and edges $e_{i,j}$ that the robot can traverse. During execution, robots can often deviate by some distance *DIST*$_{wander}$ from the edge for a variety of reasons that include obstacle avoidance and following human tendencies to move on the right-hand side of hallways. We extend the navigation graph *NAV* by subdividing each edge $e_{i,j}$ and then extending each perpendicularly by *DIST*$_{wander}$ as shown in Figure 4.2b. We call this the extended graph *NAV*$_{coverage}$.

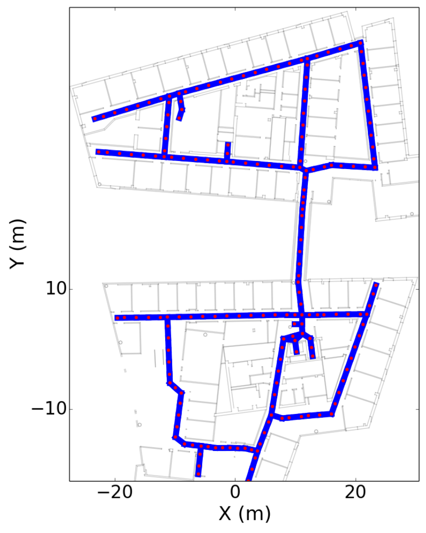
Generating Regions with Non-Overlapping Coverage

First, we generate a set of non-overlapping grids that covers only the regions that the mobile platform can reach as shown in Figure 4.2c. As shown in Algorithm 2, we achieve this by first applying a metric grid across a 2D map and then filter those that overlap with the extended navigation graph *NAV*$_{coverage}$. This takes advantage of the property that metric grids generate a set of non-overlapping grid regions.

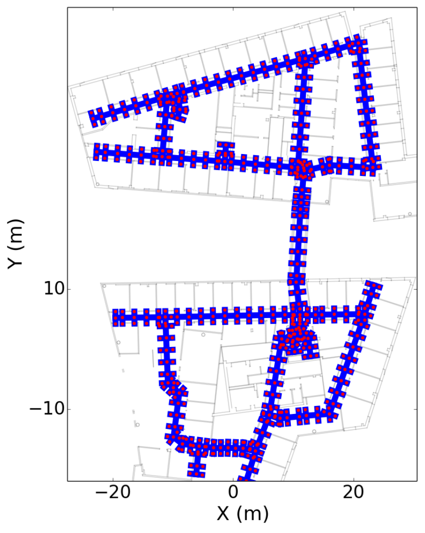
Algorithm 2 Filtering Grids that Cover Robot Navigation

```
1: $REGIONS_{cvg} \leftarrow \{\}$
2: for $x \leftarrow X_{min}$ to $X_{max}$ by $SIZE$ do
3:     for $y \leftarrow Y_{max}$ to $Y_{max}$ by $SIZE$ do
4:         $bbox \leftarrow \{(x, y, x + SIZE, y + SIZE)\}$
5:         for $edge$ in $NAV$ do
6:             if $isIntersect(edge, bbox)$ then
7:                 $reg \leftarrow (x, y, x + SIZE, y + SIZE)$
8:                 $REGIONS_{cvg} \leftarrow REGIONS_{cvg} + reg$
9:             end if
10:        end for
11:    end for
12: end for
13: return $REGIONS_{cvg}$
```

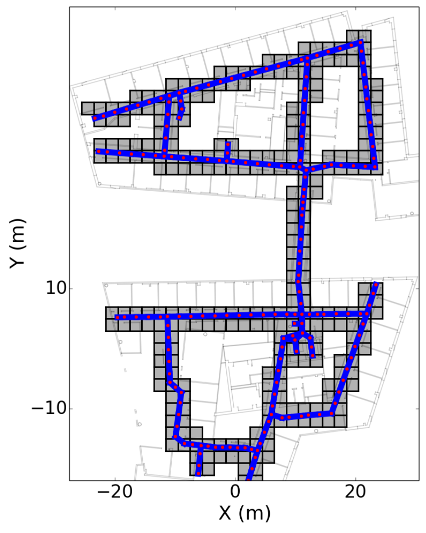
Define a fixed metric grid as the number of columns correspond to the number of non-overlapping square of size *SIZE* between (X_{min}, X_{max}) and similarly for rows (Y_{min}, Y_{max}).



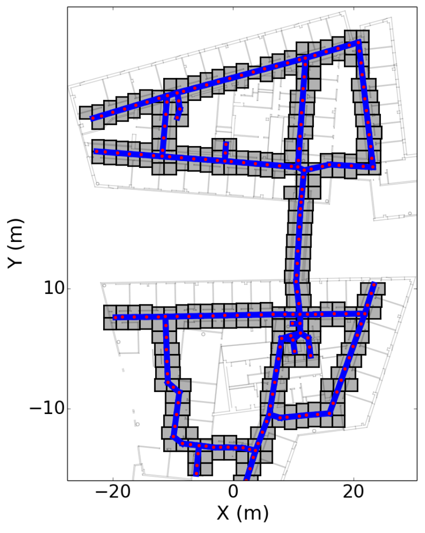
(a) Nav. Graph



(b) Extended Nav. Graph



(c) Fixed Grid



(d) Nav. Adjusted Grid

Figure 4.2: Generating navigation adjusted grids from the navigation graph.

Define each square grid cell $c_{i,j}$ as denoted as the corresponding row and column, which corresponds directly to a region r_k. For example, region r_0 corresponds to grid cell $c_{0,0}$ with bounding box $\{X_{min}, Y_{min}, X_{min} + \textit{SIZE}, Y_{min} + \textit{SIZE}\}$. Starting from (X_{min}, Y_{min}), we keep grid regions that intersect with any edge in the navigation graph *NAV*. We refer to this set of regions as $\{r_0, r_1 ... r_m\} \in \textit{REGIONS}_{coverage}$. The problem with $\textit{REGIONS}_{coverage}$ is that all grid cells are aligned to the initial grid position (X_{min}, Y_{min}) and grid cell size *SIZE*. As a result, there are likely to be many grid cells that just barely cover some reachable areas and result in uneven distribution of measurement samples.

Grid Adjustment Options

The next step is to re-position these grid cells so that they are better aligned with reachable areas as shown in by $\textit{NAV}_{coverage}$. We want to ensure that these adjustment retain the properties of avoiding region overlap while maintaining coverage over all reachable areas. Our approach first determines which groups of cells can be adjusted, which direction they should be shifted, and then applies the adjustments. We introduce an incremental approach that repeatedly chooses and commit a set of cells to adjust as described below.

Algorithm 3 Identify Grid Cell Adjustment Direction

```
1: $\textit{AMATRIX} \leftarrow \{\}$
2: for $i \leftarrow 0$ to $N$ do
3:     for $j \leftarrow 0$ to $M$ do
4:         $nhoriz \leftarrow GetNHorizontal(\textit{CMATRIX}, i, j)$
5:         $nvert \leftarrow GetNVertical(\textit{CMATRIX}, i, j)$
6:         if $nhoriz == nvert$ then
7:             $\textit{AMATRIX}[i][j] \leftarrow Both$
8:         else if $nhoriz < nvert$ then
9:             $\textit{AMATRIX}[i][j] \leftarrow Horizontal$
10:        else
11:            $\textit{AMATRIX}[i][j] \leftarrow Vertical$
12:        end if
13:    end for
14: end for
15: return $\textit{AMATRIX}$
```

The first step is to identify which group of grid cells can be shifted together. The idea is that cells need to be shifted as a contiguous group. Two adjacent cells that share a border must be shifted together or else we risk creating coverage gaps that do not fully cover the reachable areas. We consider cell adjustments in either vertical or horizontal directions. Once shifted in a particular direction (e.g. vertical), we can no longer shift them in the other direction (e.g. horizontal) as it creates the potential for overlapping coverage. Given this constraint, we encourage more adjustments by trying to adjust smaller contiguous sequences of cells. This is implemented in

Algorithm 4 Shift Vertical Grids Cells

```
1:  $VertSets \leftarrow \{\}$
2:  $VertCells \leftarrow \{\}$
3:  for $i \leftarrow 0$ to $N$ do
4:      for $j \leftarrow 0$ to $M$ do
5:          if $AMATRIX[i][j] == null$ then
6:              $VertSets \leftarrow VertSets + VertCells$
7:              $VertCells \leftarrow \{\}$
8:          else
9:              $VertCells \leftarrow VertCells + (i, j)$
10:         end if
11:     end for
12: end for
13: for $VertCells \in VertSets$ do
14:     $ys_{int} \leftarrow GetYIntersects(VertCells)$
15:     $ys_{cells} \leftarrow GetYCoords(VertCells)$
16:     $y_{int} \leftarrow (max(ys_{int}) + min(ys_{int}))/2$
17:     $y_{cells} \leftarrow (max(ys_{cells}) + min(ys_{cells}))/2$
18:     $y_{shift} \leftarrow y_{int} - y_{cells}$
19:     $RepositionY(VertCells, y_{shift})$
20: end for
21: return $DMATRIX$
```

Algorithm 3 where we create an adjustment matrix *AMATRIX* that assigned each grid cell a direction (horizontal or vertical) based on the smaller number of contiguous cells in each direction.

The next step is to identify which groups of contiguous grid cells to adjust. We cannot arbitrarily adjust cells in any direction so we look for contiguous grid cells that agree on the same direction in the adjustment matrix *AMATRIX*. Algorithm 4 shows how we search for vertical adjustments by first identifying contiguous sequence of cells $VertCells$ that agree on direction as shown in lines 4-11. Next, we shift $VertCells$ around the mean of the most extreme reachable y-axis points of $NAV_{coverage}$ covered within the set of cells. After the set of cells in $VertCells$ has been adjusted, they can no longer be re-positioned so we nullify the direction of each cell in the adjustment matrix *AMATRIX*. We also nullify cells diagonal to the two endpoint cells as it create potential overlaps. Over time, the number of readjustment options shrinks as cells are re-positioned. Once no options remain, we reach the final NAG that is much better aligned to the underlying reachable areas.

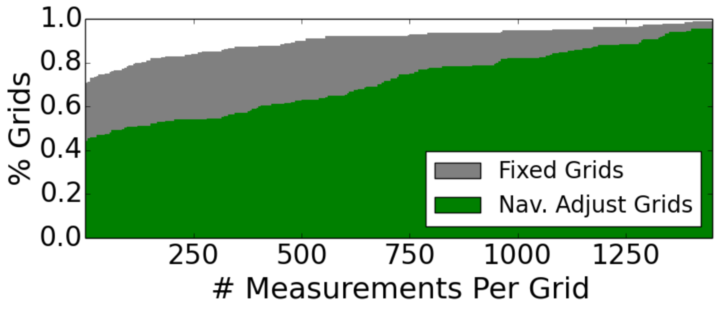
4.2.4 Evaluating Navigation Adjusted Grids

Opportunistic data collection wherever the robot happens to go will inevitably create highly skewed measurement coverage in practice. To address these fundamental limitations when collecting measurements with a mobile platform, we need to better make sense of sensor data and be more pro-active about where sensor measurements are collected. Our results here will show that NAG's lead to better distribution of sensor data across discrete regions in our environment. Better distribution ensures that each grid region has more sensor measurements so that we have more samples to compute more statistically significant insights. Chapter 5 will discuss how we can generate more intelligent navigation trajectories that effectively incorporate sensor measurement priorities.

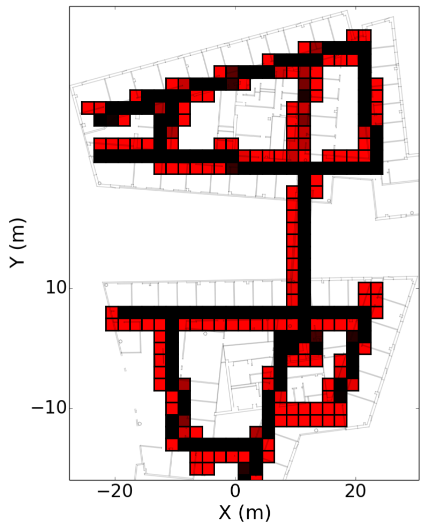
Even if we instruct the robot to execute the same coverage cycle over a long period time, there is still biased measurement coverage due to navigational constraints of the environment (e.g. bottleneck bridge more often crossed versus hallway dead-end). Our NAG's better spatially segment the environment because regions are better aligned to the robot's navigation graph. This is important because the effectiveness of discrete efforts depends on the effectiveness of their discrete boundaries.

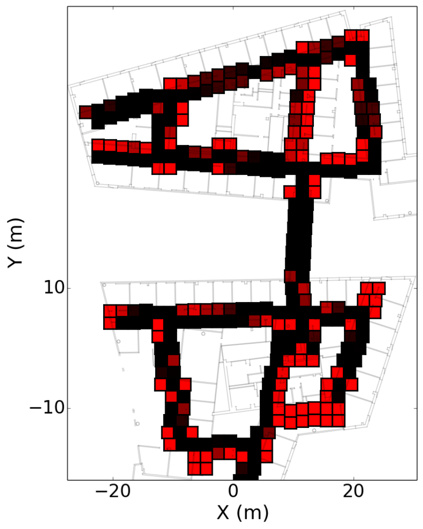
Figure 4.3a shows the distribution of measurements per region when using fixed grids versus NAG. These measurements were collected during the 5 week deployment in GHC. There are a total of 343 square grids for both representations and over 900,000 wireless measurement samples. Notice that NAG shows a 20% larger proportion of measurements per region with non-zero measurements despite using the same number of measurements. This suggests the NAG results in significantly fewer regions with little coverage of the robot's reachable locations.

We can further see how the distribution of measurement samples spatially differs. In Figure 4.3b and 4.3c, strong red grid cells indicates 0 measurements while strong black indicates at least 1500 samples. We notice poor measurement distribution is ubiquitous across all hallways (partially due to the robot's impressive precision while navigation) and the NAG helps to significantly improve measurement distribution. Clearly, there are still numerous locations with very few sensor measurements that more adaptive data collection trajectories in Chapter 5 will help to address.



(a) CDF Measurements Per Grid





(b) Fixed Grid Heatmap (c) NAG Heatmap

Figure 4.3: Comparing distribution of sensor data when using fixed grids versus NAG.

4.3 Measurement Scoring Metrics

With a discrete representation, sensor measurements are grouped according to spatio-temporal context. We will be able to apply a wide range of traditional statistical techniques to each group of measurements without needing to consider the dimensions of time and location. This results in fast computations to characterize the behavior of each group so that we can direct our focus to differentiating spatio-temporal trends across groups. We first discuss metrics that can be applied to groups of measurements. We then show how to analyze spatio-temporal trends across different groups.

Recall that our discrete representation groups sensor measurements into a data matrix bin $b_{i,j}$ composed of a set of collection records w_k. Recording data as collection records is a subtle but significant decision as specifying start and end times for each record w_k enable us to support time-dependent metrics (e.g. number of samples based on total visit time). This is important because a mobile platform does not uniformly distribute its data collection time so metrics that incorporate the amount of time spent are valuable. We introduce several types of simple scoring metrics one can use to characterize a group of measurements.

Total Visit Time Total visit time is the sum of differences in end and start time across all visit records:

$$t_{visit}(b_{i,j}) = \sum^{w_q \in b_{i,j}[w_{list}]} w_q[\mathit{end}] - w_q[\mathit{start}] \tag{4.1}$$

Number of Visits The number of unique visits:

$$n_{visit}(b_{i,j}) = len(b_{i,j}[w_{list}]) \tag{4.2}$$

Visit Frequency We define the visit frequency to be the proportion of data acquisition opportunities dedicated to this particular grid slice compared to the entire deployment time:

$$f_{visit}(b_{i,j}) = \frac{t_{visit}(b_{i,j})}{\sum_0^n \sum_0^m t_{visit}(b_{n,m})} \tag{4.3}$$

Data Variance Variance can be directly computed as:

$$d_{var}(b_{i,j}) = var(\bigcup_0^{w_q \in b_{i,j}[w_{list}]} w_q[s_{list}]) \tag{4.4}$$

Data Entropy Sensor data uncertainty can also be used to differentiate measurement value. One can approximate entropy using a histogram-based approach, where p_i is the frequency of unique sensor value k.

$$d_{entropy}(b_{i,j}) = \sum_0^{k \in set(b_{i,j}[w_{list}][s_{list}])} -p_k log(p_k) \tag{4.5}$$

Notice that there are generally two types of metrics: measurement coverage metrics and data-specific metrics. Measurement coverage metrics include total visit time, number of visits, and visit frequency that are computed without looking at sensor values. In contrast, data-specific metrics focus explicitly on the sensor values in order to compute features like data variance and data entropy. Regardless of what metrics we wish to compute, our collection records w_k define the data in a way that allows us to quantify groups of sensor measurements in many different ways.

Benefits of Quantifying Sensor Measurements Applying these scoring metrics across our discretized sensor measurements gives flexibility when attempting to extract insights about our environments. There are many different techniques one could apply so let us present a simple representative example. For this example, our goal is to understand how wireless signal propagation for an access point differs in space. Recall that wireless signals generally do not propagate idealistically due to various environmental effects (structure, materials, etc.). The idea is to apply clustering as a scoring metric to see which regions do in fact follow idealistic signal propagation models.

Raw wireless measurements are shown in Figure 4.4a, where we are not able to draw many conclusions from the visualization. We can only conclude that RSSI decreases as one moves farther away from location (-10,5), where the access point is located. Next, we might attempt to cluster regions together based on signal strength as shown in Figure 4.4b. Unfortunately, we can draw few insights from these clusters as they have little spatial coherence. This leads to the clusters shown in Figure 4.4c, where we utilize wireless domain knowledge about wireless signal propagation. These clusters combine regions based on how well they fit parameters of the signal propagation model. Notice that the eventual clusters follow know trends where signal generally propagate similarly along hallways and then diverge at intersection.

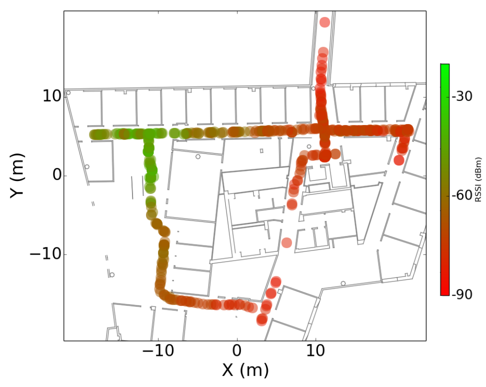
To identify clusters, we use region growing techniques borrowed from image segmentation Adams and Bischof [1994] but instead of comparing RGB values, we compare fit with a given sensor model. The idealistic wireless signal propagation used is:

$$\begin{aligned} RSSI(loc_{sample}) = & \\ & \alpha * 10{*}log(dist(loc_{ap}, loc_{sample})) + \beta \end{aligned} \tag{4.6}$$

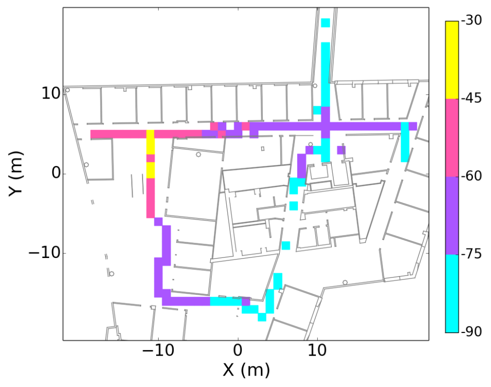
$RSSI$ for some AP at (x,y) location loc_{sample} is a function of the path loss exponent α, distance between the AP's location loc_{ap} and measurement location loc_{sample}, and some constant β. The sensor measurements provide us location loc_{sample} and $RSSI$ so the unknown parameters are location of the AP loc_{ap}, path loss exponent α and constant β.

During region growing, we incrementally combine grid measurements to find the best fitting parameters and then compute standard deviation of the fit. The resulting clusters are shown in Figure 4.4c where different groups of grids indicates different sets of signal propagation behaviors. Sensor measurements belonging to the brown cluster are shown in Figure 4.4d. Measurements for a single grid are highlighted in green, which shows that only by combining measurements across multiple grids can we robustly identify reasonable parameters for the model. To show the importance of fitting models for each group of regions, we also overlay all other wireless measurements in black. Notice that attempting to fit a single model to all of this wireless data would have resulted in a much worse and noisier fit.

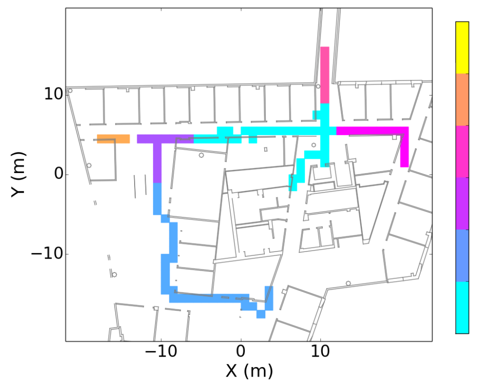
The key takeaway is that our discretization of the environment enables us to apply a variety of scoring metrics so that we can better understand how our environments vary over individual slices of both time and space. In this section, we showed that one can even perform operations across groups of grids to extract coarse-grain insights. We believe there is great potential to perform these types of data analysis techniques as sensor measurements become more widely available in the future.



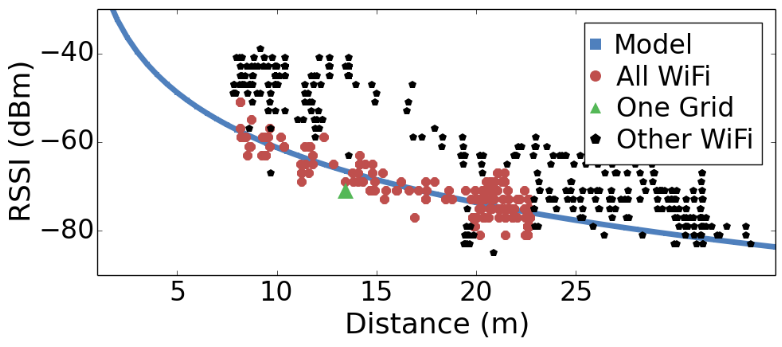
(a) Raw Measurement Heatmap



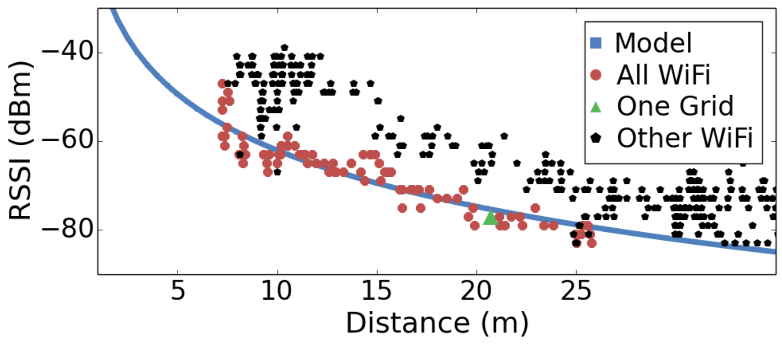
(b) Clustering by RSSI



(c) Clustering by Propagation Model



(d) Cluster 1



(e) Cluster 2

4.4 Spatial and Temporal Insights

We wish to show that fine-grain sensor measurements collected by mobile platforms can reveal valuable insights about our surrounding environments. Using data collecting from long-term sensor gathering efforts with our autonomous robot, we will be able to understand the spatio-temporal variations of our buildings. We first present results collected by the wireless monitor and then the indoor climate sensors.

4.4.1 Wireless Measurements

We first analyze wireless measurements spatially and then temporally. Recall that RSSI of wireless signals depends on the location of the transmitting device. Although the total number of wireless measurements collected is large, the number of samples become smaller as we direct our focus towards specific wireless devices. Our analysis will focus on wireless access points (APs) because they form the backbone of wireless connectivity for users and they are permanently fixed so we do not need to consider the possibility of movement. Our spatial insights will show how the wireless medium is being used by our wireless networks and then show how they actually are re-configured change over time.

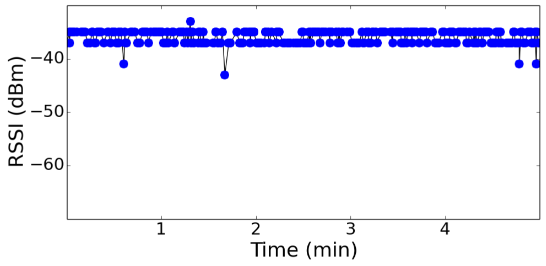
Spatial Insights

Our spatial analysis of wireless measurements reveals valuable information about how the wireless spectrum is being managed and allocated by the wireless network. First, we try to understand how wireless signals vary across our environment and what we should expect to see. Next, we show how the wireless network is serving wireless users in the environment by looking into AP coverage, channel allocation, and throughput samples. Our results will show that sensor measurements collected by autonomous robots can be processed to reveal extremely fine-grain insights about our wireless networks that our own administrators do not have available.

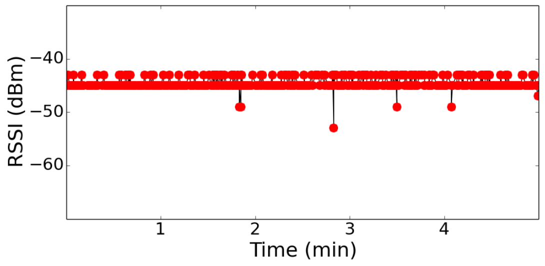
Wireless Signals The received signal strength indicator (RSSI) measures the received energy of incoming wireless packets. We begin by trying to understand how RSSI varies across our environment. We look at how RSSI varies while stationary, whether or not variance depends on mean RSSI, and how RSSI relates to throughput.

We first show wireless measurements under idealistic situations when continuously collected at the exact same location. RSSI values generally fall within the range of -90 dBm (weak signal) to -20 dBm (strong signal). Figure 4.5 shows stationary RSSI measurements captured when the WiFi device at four different distances from the AP. These measurements were captured by the robot in a hallway with no people walking around. There are intermittent outlier measurements but typically RSSI varies by no more than 2 dBm when stationary.

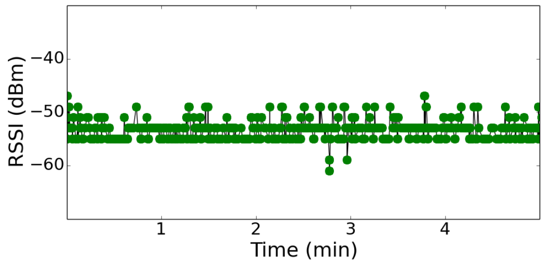
Next, we investigate how mean RSSI affects variance. This will show if high RSSI might result in lower variance. By using our discretization of the environment into 1m x 1m grid



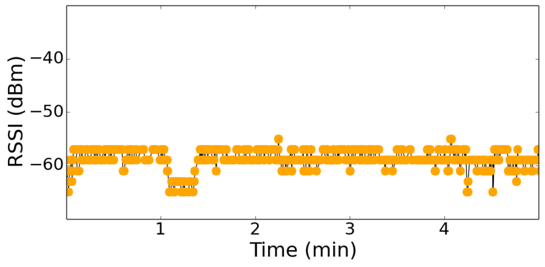
(a) 0 meters (-36.07 dBm ± 1.23)



(b) 5 meters (-44.42 dBm ± 1.15)



(c) 10 meters (-53.51 dBm ± 1.70)



(d) 15 meters (-58.8 dBm ± 1.74)

Figure 4.5: Stationary RSSI measurements at different distances from transmitting device.

regions, we can see in Figure 4.6 how that there is little correlation between mean RSSI and variance. The same trends are observed for additional APs. These results suggest that stronger wireless signals are not any less variable than weaker ones.

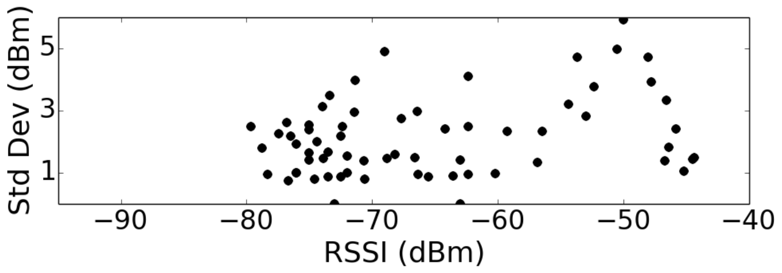


Figure 4.6: Mean RSSI versus RSSI variance for each 1m x 1m grid

Next, we show the impact of information loss due to the discretization of our environment. Figure 4.7 further inspects RSSI variance across the same 1m grid regions for all observed APs after the 5 week deployment of our robot across four floors. The average standard deviation for each floor is 2.88, 2.82, 2.94, and 2.78 dBm for GHC6, GHC7, GHC8, and GHC9, respectively. Notice that RSSI variance is higher than those measured while stationary because we are grouping measurements within 1m grid regions.

Finally, we show how RSSI relates to throughput while moving. This is a unique benefit of mobile platforms because wireless performance while moving faces unique challenges. Figure 4.8 shows that higher RSSI is only weakly correlated with higher throughput. Throughput is measured as the useful number of megabits successfully transferred per second (ignoring over-

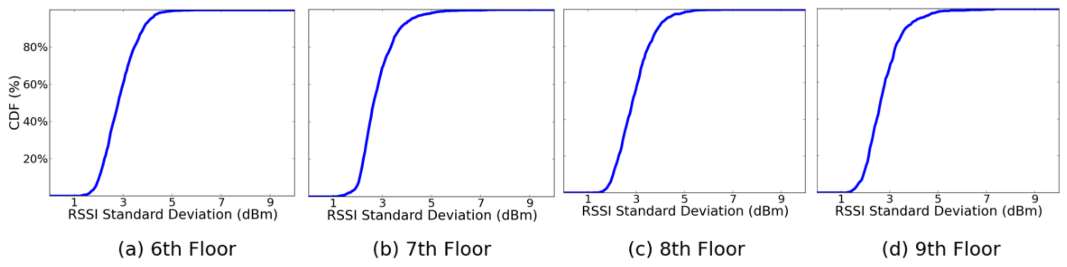


Figure 4.7: CDF of RSSI variability for each 1m x 1m grid across all APs.

heads like packet headers) as samples using the iperf tool (TCP). We can see that ensuring RSSI remains above -80 dBm is needed to ensure reasonable wireless performance. The key takeaway is that high RSSI is a pre-requisite for high throughput but not a guarantee of strong wireless performance. Chapter 7 will show in more detail what causes poor wireless performance while moving.

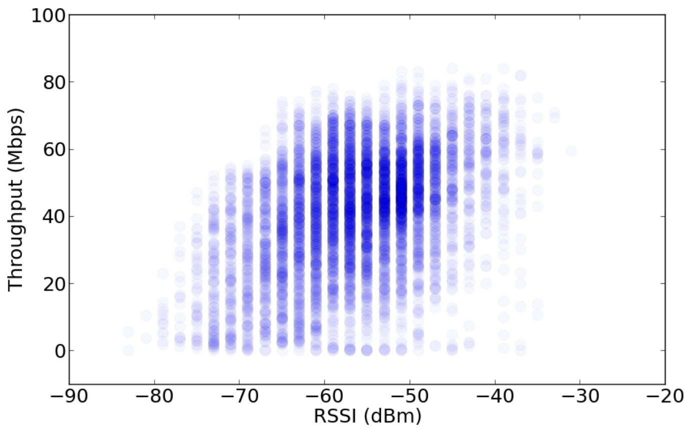
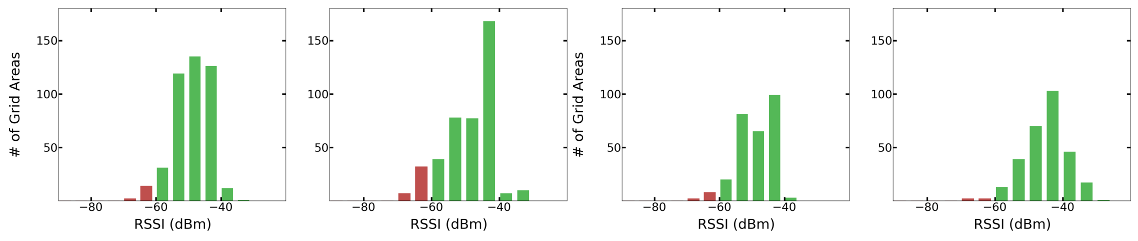


Figure 4.8: Heatmap to show RSSI versus throughput.

AP Coverage Recall that the goal of strong AP coverage is to ensure that every location has at least one high RSSI AP available. Strong AP coverage is a big concern for wireless network administrators who devote a lot of time and effort to resolving any wireless dead zone issues. Wireless measurements collected by a robot reveals much more fine-grain AP coverage insights than our own administrators can gather due to their reliance on manual efforts. Using our discrete representation, we will be able to both quantify and show the distribution of RSSI across our environment.

Figure 4.9 quantifies the distribution of the highest RSSI AP across each grid of our environment where the robot could reach. We can see that our environment has strong coverage throughout. While our own network administrators target a minimum of -60 dBm, our results show that our building has at least -70 dBm at all locations which is quite strong. Due to our efforts, we are able to quantitatively show that our building has strong AP coverage across all floors and that wireless dead zones are not a significant problem.



(a) GHC6 (b) GHC7 (c) GHC8 (d) GHC9

Figure 4.9: Histogram showing median RSSI of the best AP across four floors.

The sensor measurements collected can provide alternate perspectives of AP coverage. Figure 4.10 illustrates RSSI spatially where green indicates high RSSI and red indicates low RSSI of the highest median RSSI for each 1m grid region. With this perspective, we are able to see where the few low RSSI locations occur. We can see low RSSI situation are found primarily along the bridge where few wireless users are found.

Due to these fine-grain sensor measurements captured by our robot, we are able to view AP coverage from multiple perspectives and better understand the availability of our wireless network.

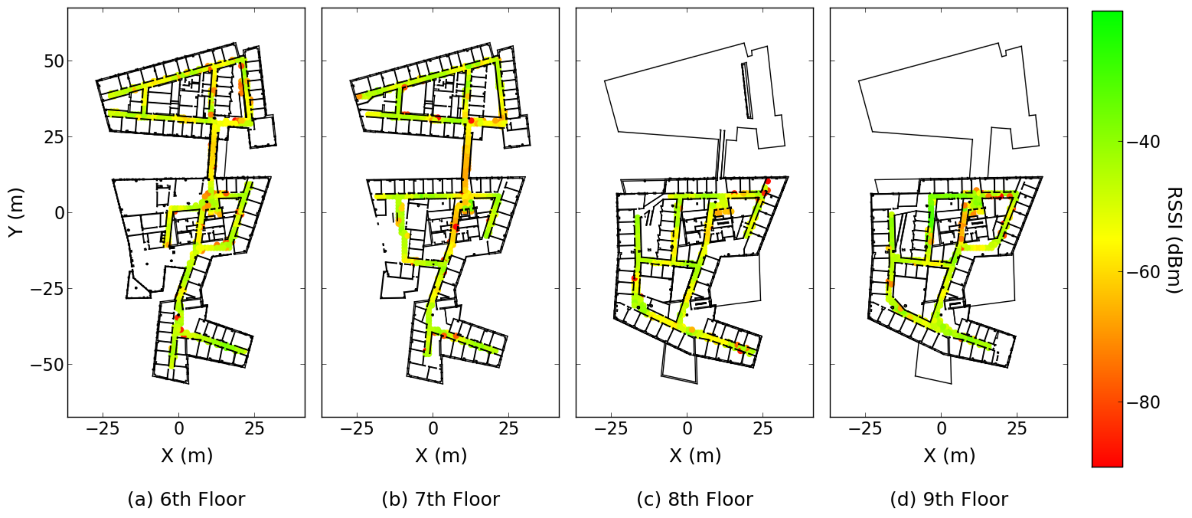
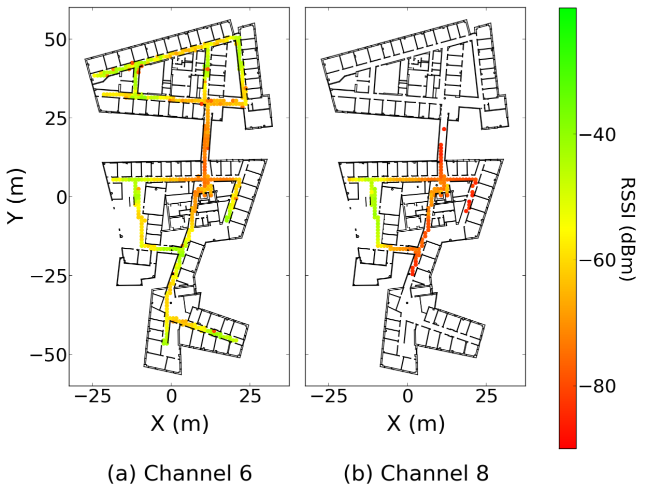


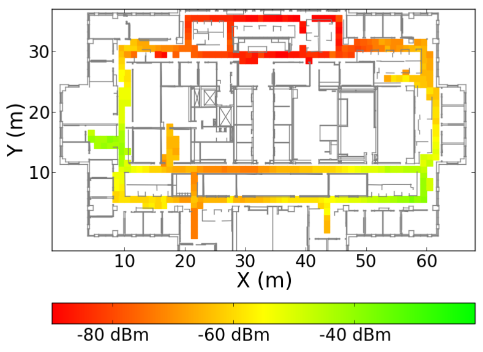
Figure 4.10: Spatial AP coverage across the four floors.

Channel Assignments Wireless measurements can be further segmented along different parameters like wireless channels to shed light on how the wireless spectrum is being allocated. Recall that each AP can be assigned to one of several wireless channels (11 channels for 2.4 Ghz and 23 channels for 5 Ghz) and the wireless network has the ability to adjust these channel assignments. Wireless channel assignments are important because a potential cause of poor wireless performance is interference due to simultaneous wireless transmissions on the same frequency in overlapping areas. A general solution is to partition the environment to minimize overlapping regions using the same wireless channel. Unfortunately, these partitioning decisions are often made by combining coarse wireless samples (collected from APs throughout the environment Reis et al. [2006], Mahajan et al. [2006]) with simple wireless models that have unknown fidelity to actual wireless conditions of the environment. With fine-grain wireless measurements collected by a robot, we can directly show exactly how wireless conditions for each channel vary over space.

In Figure 4.11, we show several wireless channels across two environments. We can see several RSSI trends regardless of the environment. First, concentrations of strong RSSI (as indicated in green) help to suggest where APs may be located. We can easily guess how many unique APs there are for each channel: seven in GHC on channel 6, one in GHC on channel 8, and two in NYU on channel 48. Notice that signal propagation is not purely a function of distance as hallways appear to influence the range of wireless signals. While typically difficult to predict in realistic environments, we are no longer faced with these issues since we have actual wireless measurements collected across numerous locations.



(a) Channels 6 and 8 for GHC 7.



(b) Channel 48 at NYU.

Figure 4.11: Comparison of RSSI trends across different wireless channels.

It is somewhat strange that there are many APs assigned to channel 6 while only a single AP on channel 8. It is recommended that APs are assigned to either 1, 6, and 11 for 2.4 Ghz because they are non-overlapping (simultaneous wireless transmissions without interference). While the APs on channel 6 are cooperatively using the wireless spectrum, the rogue AP on channel 8 introduces potential interference for nearby wireless transmissions on both channel 6 and 11, which is an inefficient usage of the limited wireless spectrum. With our wireless maps, we are able to expose these rogue APs and even help to pinpoint where it is located. Without the map, our own network administrators would be using RSSI gradient tracking tools on site.

Throughput Samples Finally, we highlight the unique opportunity to explore sensor variations that arise with motion. Recall that high RSSI is only a pre-requisite for good wireless performance while throughput reflects the rate of data transfer. We already showed that there is strong AP coverage across our building but Figure 4.12 shows that median throughput for each 1m grid region is highly variable and low throughput overall with the exception of a few select hallways. While stationary wireless performance suggested strong throughput at different locations in the environment, throughput measurements collected while moving demonstrate clearly poor wireless connectivity while moving. The opportunity to quantify sensor measurement variations while moving is a unique advantage when using robots for data collection.

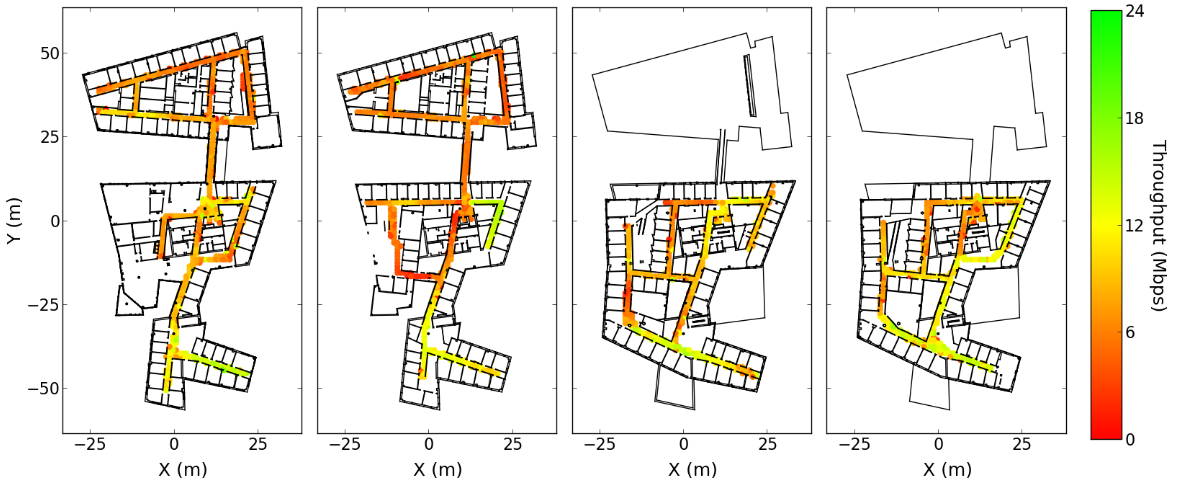


Figure 4.12: Throughput samples captured by a mobile platform while continuously moving.

Temporal Insights

We show how wireless measurements can help to reveal how the wireless medium changes over time. While our infrastructure APs remain stationary, the wireless network can re-configure their transmit power and channel assignments, which ultimately affects the wireless conditions across our environment. By tracking changes across fine-grain wireless maps over time, we will be able to spatially identify which APs are reassigned to different wireless channels over time and infer why these changes are being made.

Figure 4.13 shows weekly snapshots of RSSI on channel 11 over four weeks. Notice how the number of APs begin at 6 in week 1, reduces to 5 by week 3 and finally 3 in week 4. These changes are likely a response to the rogue AP on channel 8 in Figure 4.11a. By transitioning APs on channel 6 and 11 away from the overlapping region, the network is able to better use the WiFi medium.

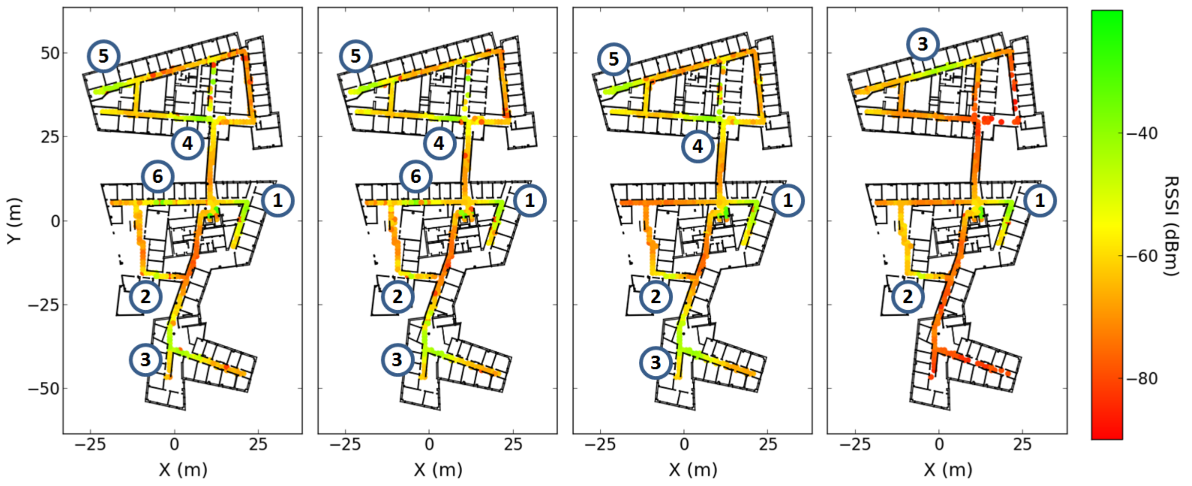


Figure 4.13: Evolution of APs on channel 11 for GHC over four weeks.

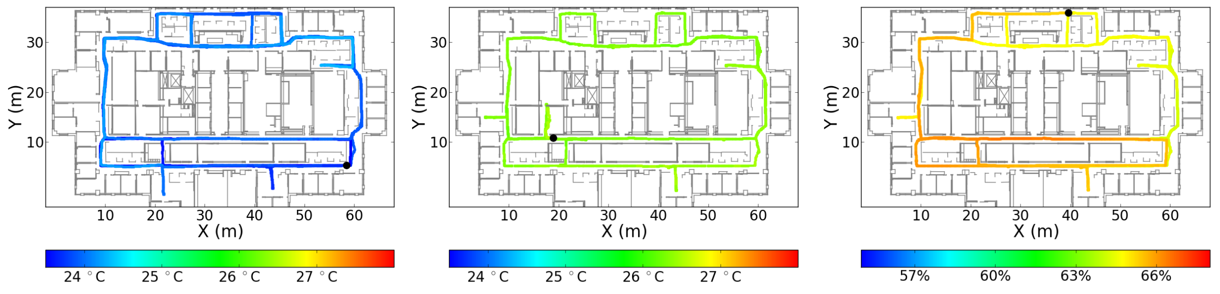
4.4.2 Climate Measurements

We now turn our attention to analyzing climate sensors both spatially and temporally. We collected climate measurements simultaneously with the wireless measurements discussed above and will perform similar discrete analysis. Temperature and humidity values fundamentally differ from wireless signal strength so our analysis will naturally focus on segmenting the measurements in different ways to reveal interesting behavior across our environment. We show that while climate measurements have little variations in space, there are fairly significant changes over time.

Spatial Insights

We wish to first highlight the value of knowing the spatial context of climate measurements. We previously showed in Figure 3.9 how temperature and humidity vary as the sensor is moved to different locations. The same measurements could have been collected without the help of a robot (although perhaps not with the same accuracy and precision). However, we are missing the spatial context that allows us to see how measurements at these different locations should be correlated. Fortunately, the data collection with mobile platforms excel at adding accurate and precise spatial context for each sensor measurement.

Figure 4.14a shows several 30 minute snapshots of fine-grain temperature and humidity maps collected across our environment. We can see that there are no major variations in sensor values across different locations, which suggests that temperature and humidity is fairly consistent across a building. We do see clear differences across snapshots making it seems like trends over time should yield interesting insights.



(a) Morning temperatures (b) Evening temperatures (c) Humidity

Figure 4.14: Indoor climate maps showing temperature and humidity from 30 minute snapshots.

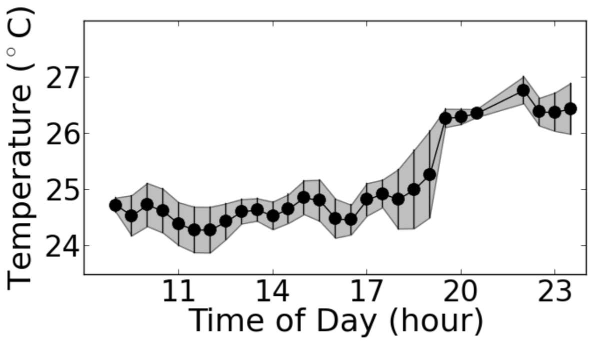
Temporal Insights

Even though a single robot has limited visibility of the environment at any point in time, the accumulation of measurements over time provides valuable temporal insights. Because climate sensors exhibit consistent periodic variations based on the time of the day, we can aggregate measurements from deployments over many days and extract fine-grain temporal insights. Using climate measurements collected during a 3 week deployment at NYU, we discretely characterize climate trends based on the time of the day. We are able to show consistent rise in temperature and humidity starting around 6 pm that were later found to be intentional efforts in turning off the HVAC system for energy savings.

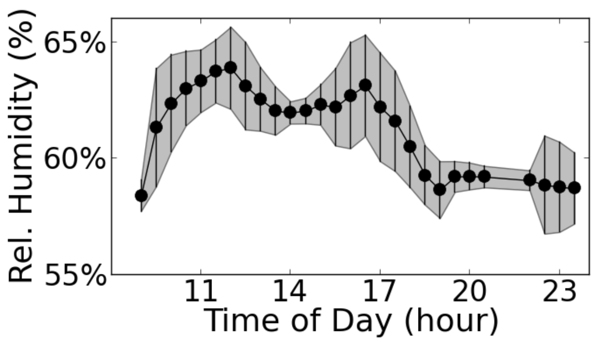
We discretize our environment spatially into 20m grid regions and temporally into 30 minutes slots over different times of the day. We begin by comparing average temperature and humidity collected by the AM2315 sensor as shown in Figure 4.15. The most significant variation occur consistently at 6 pm with simultaneous increases in temperature and decreases in humidity, which is associated with turning off the HVAC system. More interestingly is that while temperature remains fairly consistent during the work day, humidity experiences more significant variations suggesting that the HVAC system optimizes primarily for temperature alone. Interestingly, the comfort level humans feel is actually a combination of temperature and humidity so this may explain why occupants in certain offices felt colder than others.

The variations in humidity might be explained either by measurements collected across different days or across different locations. We first further segment our measurements based on the day of capture as shown in Figure 4.16. Notice that temperatures are pretty consistent across different days. In contrast, humidity differs considerably in some cases by nearly 3-4 percent depending on the date of capture. Clearly, our indoor climates follow the same general trend across different days but humidity stands out as varying the most.

Finally, we consider the different in climate based on region by simply separating the measurements based on the 20m x 20m regions uniquely numbered in Figure 4.17. In this case, we affirm our previous observations that there are few obvious differences in climate based on location. Looking at the trends a bit more carefully, we can see minor differences. For example, region #10 tends to have consistently higher temperatures than region #9 suggesting minor

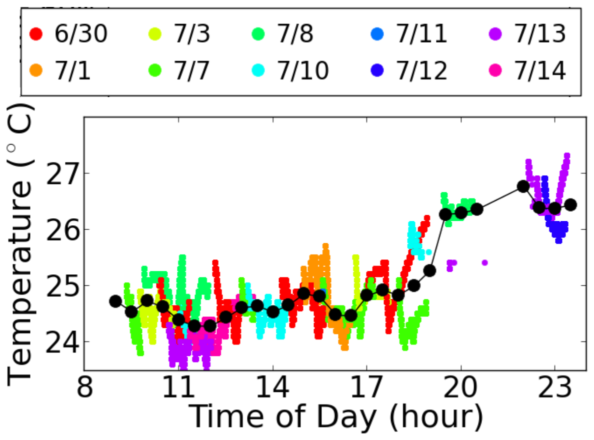


(a) Overall average temperature

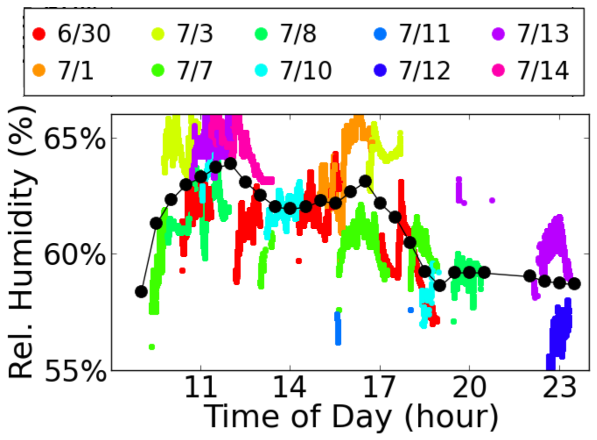


(b) Overall average humidity

Figure 4.15: Average temperature and humidity within one standard deviation in NYU.



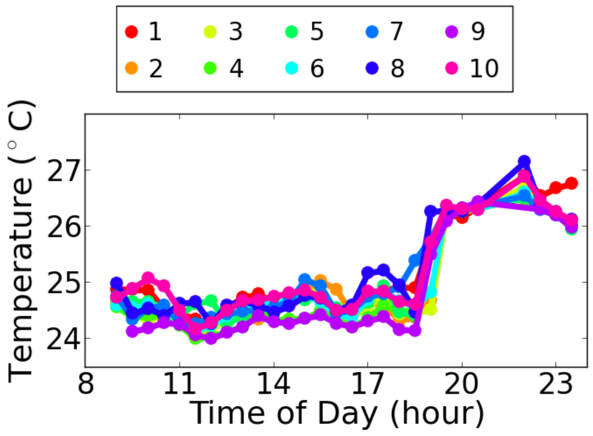
(a) Temperature



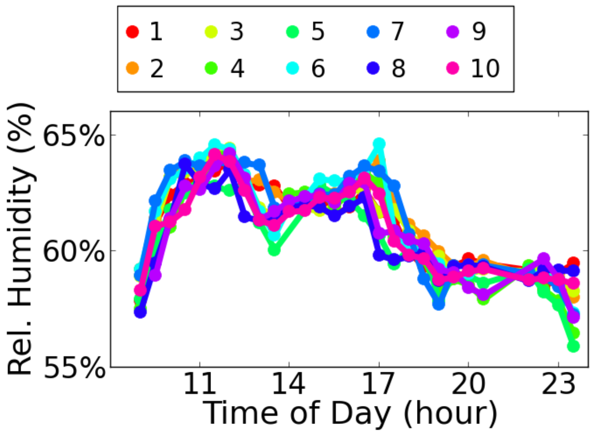
(b) Humidity

Figure 4.16: Raw temperature and humidity values separated based on measurement date.

spatial differences.



(a) Temperature



(b) Humidity

Figure 4.17: Average temperature and humidity of non-overlapping 20m grid regions.

With discrete analysis over data collected by a robot, we are able to first reveal meaningful trends in time. By further segmenting sensor measurements based on when and where they were collected, we are able to better understand possible reasons to explain where differences might arise across our environments.

4.5 Summary

In this chapter, we introduced a discrete representation to extract spatial and temporal insights from the stream of fine-grain sensor measurements captured by a mobile platform. We specifically developed a discrete spatial discretization centered around locations where the robot can actually reach for more effective distribution of sensor measurements captured while moving around. With measurements grouped based on similar spatio-temporal contexts, we could easily apply simple statistical metrics to differentiate trends across our buildings. We show how these efforts reveal insights from actual sensor data to better see exactly how our buildings vary in space and change over time.

Chapter 5

Active Navigation of Robots for Measurement Needs

Autonomous robots are unique because we can control their movements and pro-actively seek out subsequent sensor measurement locations. Where and when the sensor measurements are collected depends entirely on where the robot navigates. Relying on measurements opportunistically captured at locations traversed when executing traditional robot service tasks results in a skewed incomplete measurement coverage. We present two approaches for generating trajectories when the robot has limited data collection time. Our first approach called navigation-based trajectories focuses on making data collection efforts indiscernible from normal robot navigation. Our second approach called grid-based trajectories allows the robot more flexibility to wander along hallways in a zig-zag manner that clearly indicate the robot is focused on gathering sensor measurements.

Both approaches need to effectively utilize the robot's limited data collection opportunities by focusing its efforts on sensor measurement priorities. Because measurement priorities change over time, our algorithms will end up generating different trajectories over time. The idea is treat time as a cost and measurement priorities as a reward. By generating the highest reward path given limited time, we are able to strike a balance between collection time and sensor measurement needs. For the navigation-based trajectories, rewards are distributed along the edges of the robot's navigation graph where we find that the discounted-reward traveling salesman problem (DRTSP) algorithm is able to generate effective data collection trajectories. For the grid-based trajectories, rewards are distributed across discrete grid regions resulting in a much larger search space that we address with ant colony optimization (ACO). Our results show that these trajectories allow the robot to automatically navigate around the environment based on spatio-temporal sensor measurement needs.

5.1 Navigation-Based Data Collection Trajectories

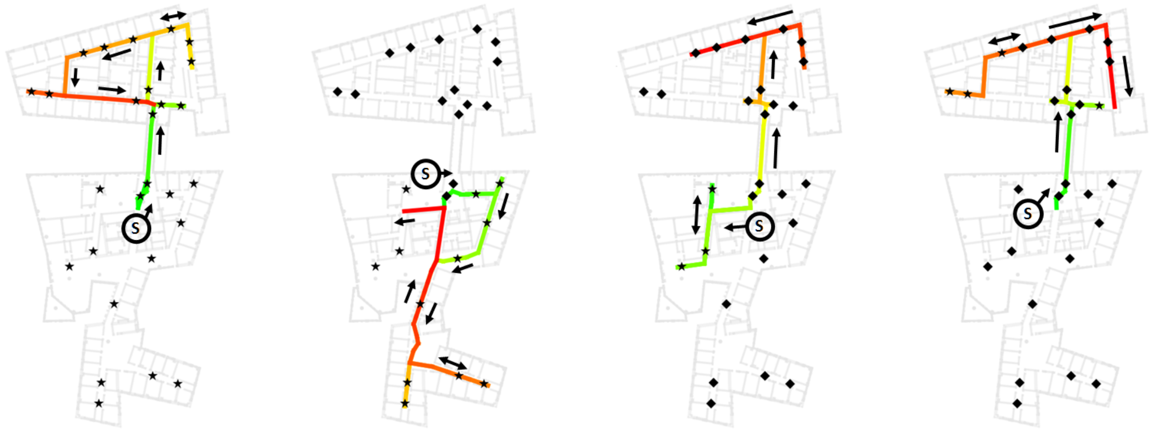
Navigation-based trajectories aim to preserve existing navigation behavior (e.g. primarily straight movements with sporadic in-place turns along the robot's underlying navigation graph shown in Figure 6.1b) so that data collection efforts are opaque to nearby humans. This is motivated by the desire for robots that operate cooperatively in the environment (e.g. moving at predictable speeds and few sudden stops).

We formulate the planning problem as a reward collecting traveling salesman problem (TSP). Even if deployed from the same starting location, different paths will be generated over time because the algorithm uses a history of data collection efforts over time to influence measurement priorities as shown in Figure 5.1. We evaluate our framework under several variations of the traveling salesman problem and find the discounted-reward TSP performs well when even faced with operational uncertainties (e.g. data collection time).

5.1.1 Path Planning for Data Collection

Our approach computes trajectories at the beginning of each active data collection deployment by combining the robot's starting location, expected deployment time, and the measurement value of different locations. Should the robot have sufficient time to complete the entire path, we simply compute another path based on the updated measurement priorities (e.g. if targeting measurement coverage, we update the time since last visit).

Because we want these trajectories to traverse naturally along the robot's underlying navigation graph, our goal is to generate paths that traverse the edges of this graph. We discriminate navigation paths by assigning costs (e.g. time to traverse) and one-time rewards (e.g. time since last visit) along each edge of the navigation graph. We use one-time rewards as they are supposed to reflect measurement priorities. Once an edge is visited, it should have lower reward so that the trajectories do not endlessly traverse the same edge. By readjusting these one-time rewards between deployments, this will enable two successive deployments from the same starting location to have completely different navigation strategies like the example show in Figure 5.1.



(a) Deployment #1 (b) Deployment #2 (c) Deployment #3 (d) Deployment #4

Figure 5.1: Different navigation path choices across multiple deployments.

5.1.2 Formulation as Prize Collecting TSP

We assume that the robot wants to visit all measurement locations if time permits. With uncertain deployment time, the robot needs to identify paths that effectively utilize the robot's time even if interrupted earlier or later than expected. Each edge e_{meas} of the robot's navigation graph is attached with a corresponding cost and reward parameters $\langle e_i, c_i, r_i \rangle$. With this representation, was can apply variants of the prize collecting traveling salesman problem to compute navigation trajectories. Greedily visiting edges based on highest measurement priority is likely to be sub-optimal because it takes time to navigate from edge to edge. An effective algorithm will balance the cost and rewards to best utilize the robot's limited deployment time.

Cost-Reward Parameters

The robot's active data collection strategy can be influenced by adjusting the cost and reward parameters.

There are two types of costs for each measurement location: navigation cost c_{nav} and measurement cost c_{meas}. The navigation cost is based on traversal time, the time to cover the shortest distance across the edge at normal speeds. Measurement cost can be much higher for reasons like slower speeds for more accurate sensor measurement locations. As a result, the total cost c_i of an measurement location is either its navigation cost or measurement cost $c_i = \{c_{nav} | c_{meas}\}$ depending on whether the robot is simply driving through or capturing measurements along the edge.

Rewards discriminate the relative importance of different measurement locations. The robot collects one-time rewards for each measurement location r_i for each deployment that are either: no reward r_{no}, low priority reward r_{low}, or high priority reward r_{high}. The total reward is simply the sum of these rewards $r_i = r_{no} + r_{low} + r_{high}$. As an example, the low priority rewards can be the difference in time from the current time and time since last visit to help bias unvisited locations. The high priority rewards provide additional flexibility to further influence the robot's navigation for particularly important measurement locations. For example, if we wish for a location to be visited once a week, then we adjust r_{high} based on this condition.

TSP for Computing Paths

With the formulation of each edge as $\langle e_i, c_i, r_i \rangle$, we can construct a tour to visit all reward-producing edges. We explain why this is suitable for variants of the prize-collecting traveling salesman problem. First, the traditional traveling salesman problem (TSP) only considers finding the shortest path cycle so it does not incorporate the different rewards across edges. The TSP is really only well-suited when the robot has enough time to visit all reward-producing edges.

$$min \sum_{0}^{\infty} c_i \tag{5.1}$$

In contrast, the prize collecting TSP (PCTSP) incorporates rewards collected over time. One variant is called the orienteering problem, which finds the maximum reward path within a fixed time limit MAX_COST. The orienteering problem only computes a path within the given time limit so it is best when the exact deployment time is known.

$$\begin{gathered} max \sum_{0}^{j} r_i \\ s.t. \sum_{0}^{j} c_i \leq MAX\_COST \end{gathered} \tag{5.2}$$

Another variant called the discounted-reward TSP (DRTSP) finds the maximum discounted reward path. Essentially, rewards are weighted with a discount rate γ depending how much cost has been accumulated up to that point. The idea is that the longer it takes to collect rewards at a certain location, the lower the expected reward should be because of the increased likelihood of the deployment being interrupted. The discounted-reward generates path traversing all reward-producing edges that may be longer than those generated by the TSP if higher cumulative rewards are collected earlier along the path.

$$max \sum_{0}^{\infty} \frac{r_i}{(1+\gamma)^{c_i}} \tag{5.3}$$

Selecting an appropriate discount rate γ is very important parameter. Equation 5.4 shows how one can select an appropriate discount rate γ by providing the proportion of reward p that should remain after some expected deployment time t. For example, in our evaluation, we aim to discount the reward by 50% of the original value by the time 20 minutes have passed. As a result, we use a discount rate of 1.2%. By adjusting the discount rate, one can influence how important it is for measurements to be collected earlier or later during a deployment.

$$\begin{gathered} \frac{1}{(1+\gamma)^t} = p \\ \gamma = e^{\frac{ln(\frac{1}{p}))}{t}} - 1 \end{gathered} \tag{5.4}$$

Solving these variants of the TSP is NP-hard but we are working with metric maps, which is the special case Euclidean TSP. As long as costs are only based on distance, we can use the minimum spanning tree heuristic in combination with the triangle inequality to find paths within a factor of two optimal Jaillet [1988]. Using this heuristic, we are able to quickly compute navigation trajectories that effectively prioritize sensor measurement needs.

Adjusting Rewards Across Deployments

Our original goal was to adjust trajectories over time in response to changing measurement needs. Notice that the DRTSP will generate the same trajectory given the same inputs. By readjusting rewards after each deployment, we can ensure that the robot reacts to changing measurement needs. For example, if we are targeting measurement coverage then rewards will be higher for less recently visited locations. By computing the DRTSP with different sets of rewards across deployments, we are able to generate intelligent and reactive navigation strategies that adapt over time.

5.1.3 Evaluation

In our evaluation, we compare the DRTSP against other variants of the TSP and show that it is best suited to handle uncertain deployment time. We first compare how different TSP variants balance the tradeoffs between cost and rewards during a single deployment. We then show how DRTSP distinguishes itself across multiple deployments because it computes navigation trajectories that react to changing measurement needs over time.

Single Deployment Path Strategies

We want to distinguish how effectively the different algorithms use the robot's limited data collection time. Presented with a specific situation, we show how much reward each approach would collect if it had been interrupted after different amounts of time.

In our example, we consider a robot with an expected deployment time of 20 minutes. From a real navigation map of an enterprise environment, we have marked 25 high priority hallway segments assigned uniform expected measurement reward value of 1. Low priority hallways can be optionally visited with a reward of .01 to incentivize traversals across different hallways.

The orienteering algorithm is provided a 20 minute limit while the DRTSP uses a discount rate of 1.2% computed from the expected deployment time of 20 minutes. Figure 5.2 shows the accumulated rewards by the three approaches. There are clear differences in how each planner prioritizes the order in which to visit these locations. First, the original TSP naturally finds a path that takes the least amount of time since it only consider the costs. As a result, it has the lowest total reward collected by the 20 minute time limit but will be the first path to reach the maximum reward. Both the orienteering problem and discounted-reward TSP (DRTSP) incorporate rewards so it is no surprise they find paths with high total reward by the 20 minute mark.

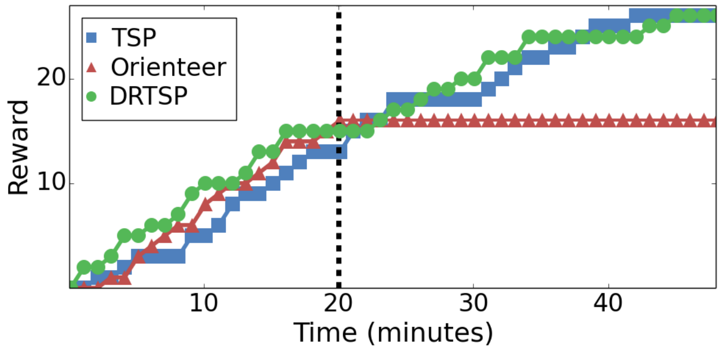


Figure 5.2: Total reward collected over time for one path by the three variants of the TSP.

The orienteering solution excels when the exact deployment time is known but has no contingency plan should there be additional time. In essence, the orienteering problem succumbs to the same limitations as greedy approaches that locally navigate to the next highest reward location without considering the effect on the remaining reward collection opportunities. Of course, the orienteering solution does have a slightly longer greedy time horizon that it considers.

In contrast, the DRTSP has a plan for additional deployment time so it will continue traversing the rest of the environment until all locations are visited. Notice that the total reward before the 20 minutes time limit remains consistently higher than the others, which means the DRTSP will collect higher rewards even if interrupted early. This arises due to the DRTSP placing a higher value on rewards collected earlier during the deployment. An additional benefit of the DRTSP is the opportunity to motivate exploration of different routes. With the small .01 reward for low priority locations, we see that DRTSP covers an additional unique 14.16 meters over the shortest path TSP in exchange for traveling an additional 33.22 meters as shown in Figure 5.3.

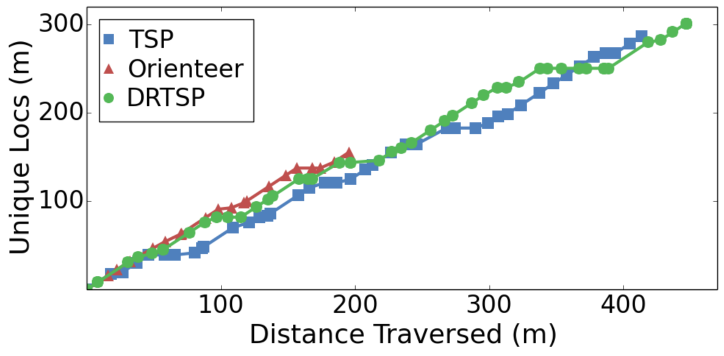


Figure 5.3: Total distance traveled versus # unique locations visited.

Across Multiple Deployments

We investigate how navigation paths evolve across multiple deployments due to readjustment of rewards after each deployment. We apply a simple situation where high priority locations are assigned reward 1 if not visited in the last three deployments and reward .1 for the remaining low priority locations.

First, we show what it means to adjust navigation strategies over time. Figure 5.1 shows how the three algorithms perform when the robot is given a fixed 20 minute time limit each deployment when starting from the same initial position. High priority locations (stars) have not been visited in the previous three deployments versus low priority locations (diamond) have been recently visited. The starting position indicated with the circle S. The navigation order is indicated by the gradient from green to red as well as select arrows. It takes at least three deployments to fully cover the environment requiring effective usage of data collection opportunities. As we can see, adjusting the rewards allows the robot to react to its previous data collection efforts.

Next, we show how effective these algorithms perform across multiple deployments with uncertain deployment times. We consider deployment times randomly generated from Gaussian distribution with mean of 20 minutes and standard deviation of 10 minutes when starting from the same initial location. Rewards are adjusted after each deployment as before. Orienteering is once again given a 20 minute time limit and DRTSP is given a 1.2% discount rate. Figure 5.4 shows the proportion of measurement locations visited within the last three deployments. It is desirable to have higher proportion of recently visited measurement locations as it indicates more thorough measurement coverage. Since the TSP does not consider rewards, it simply executes the same navigation path and serves as a baseline. The orienteering solution is able to maintain a higher proportion of recently visited locations but it fails to take fully advantage of situations where the robot has additional time. The DRTSP is most effective at ensuring fresh measurements since it emphasizes paths that collect rewards early while also having a plan in case the robot is deployed longer than the expected deployment time.

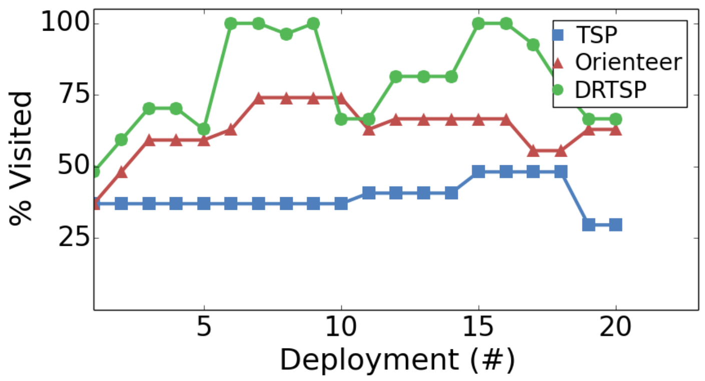


Figure 5.4: Percentage of measurement locations visited in the previous three deployments.

5.2 Grid-Based Data Collection Trajectories

If we allow the robot to more freedom to move around (e.g. abrupt stops and zig-zag motions) for targeted data collection, then we can collect higher granularity sensor measurements to better distinguish fine-grain variations (e.g. more measurements within corridors). No longer constrained by the coarse-grain navigation graph, we can generate detailed trajectories across a grid-based representation of measurement priorities. Given the larger search space, traditional variants of the TSP are unable to scale effectively so we turn to stochastic efforts like ant colony optimization for generating trajectories. Our evaluation will show that such heuristics are able to effectively utilize the robot's limited data collection time while also incorporating the spatio-temporal value of sensor measurements.

5.2.1 Trajectories for Spatial Measurement Needs

We first consider trajectories when given a spatial snapshot of measurement needs. Given a discrete spatial representation (e.g. our navigation adjusted grids) where some cells are marked with one-time rewards as shown in Figure 5.5a, a fixed deployment time, and a starting location, we are once again faced with the prize collecting traveling salesman problem (PCTSP) Kantor and Rosenwein [1992]. Known to be NP-hard and facing a large search space, we turn to the ant colony optimization (ACO) heuristic because it is an incremental algorithm that scales better than alternate efforts and quickly finds reasonable solutions Montemanni et al. [2011] like those shown in Figure 5.5b.

We first define the search space and then discuss how ACO generates navigation trajectories. Our search space is a set of vertices and edges. Each grid cell $c_{a,b}$ is vertex v_i and grid cell adjacencies correspond to an edge $e_{i,j}$. The distance cost $d_{i,j}$ is defined to be distance between midpoints of adjacent regions. Reward r_i is attached to each vertex $vertex_i$ We also have a heuristic value $\eta_{i,j}$ for each value representing the reward of reaching vertex v_j by traveling over the edge, $\eta_{i,j} = \frac{r_j}{d_{i,j}}$. We are once again faced with a search space of costs and rewards over a

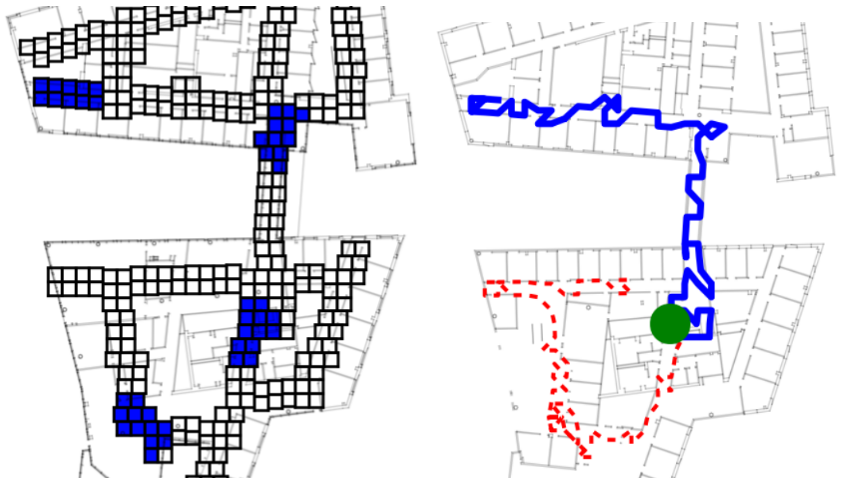

(a) Reward Snapshot (b) Two Ant Trails

Figure 5.5: Example of rewards distributed across environment and ant trails.

graph over which we wish to compute a maximum reward path.

In ACO, simulated ants move across grids in the environment (represented as a graph of vertices V and edges E), leaving phermones $\tau_{i,j}$ along the edges traversed. These phermones placed along edges of the graph provide information to subsequent ants reflecting the desirability of the trail. Each time an ant reaches a vertex v_i, it chooses between possible edges to traverse from these phermones. According to the probability of exploitation $P_{exploit}$, either the ant 1) *explores* by randomly selecting an edge based on a weighted probability of these phermones or 2) *exploits* by selecting the most desirable path. Over time, edges $e_{i,j}$ leading to higher rewards end up having more phermones $\tau_{i,j}$, which in turn encourages more ants to follow. The result of this algorithm are navigation trajectories that effectively utilize the robot's time for collecting high priority sensor measurements.

Ant Colony Optimization Algorithm

We discuss the ACO algorithm in more detail as shown in Algorithm 5. At the high level, we track the best path discovered across multiple ant colony iterations ant_{iters} as shown in line 2. For each ant colony, we create a set of ants *AntToRun* that are given an initial start location loc_{start} and collection time $time_{collect}$ as shown in line 4. The ant colony enters a loop where each ant takes turns adding an additional vertex to its path. Each ant randomly decides to either explore or exploit phermone information in line 9-12. The choice is based on a given probability of exploration $P_{exploit}$.

Algorithm 5 Ant Colony Optimization

```
1: $PhermoneMap \leftarrow \{\}\{\}$
2: $BestPath \leftarrow \{\}$
3: for $i \leftarrow 0$ to $NIterations$ do
4:     $\textit{AntToRun} \leftarrow CreateAnts(\textit{loc}_{start}, \textit{time}_{collect})$
5:     $AntsDone \leftarrow \{\}$
6:     while $len(\textit{AntToRun})$ != 0 do
7:         $ant \leftarrow \textit{AntToRun}.pop()$
8:         $decision \leftarrow random()$
9:         if $decision < P_{exploit}$ then
10:            $v_{next} \leftarrow ant.Explore()$
11:        else
12:            $v_{next} \leftarrow ant.Exploit()$
13:        end if
14:        $ant.applyLocalPhermones(v_{next})$
15:        if $ant.isDone$ then
16:            $AntsDone \leftarrow AntsDone.push(ant)$
17:        else
18:            $AntstoRun \leftarrow \textit{AntToRun}.push(ant)$
19:        end if
20:    end while
21:    for $ant \in AntsDone$ do
22:        $ant.applyGlobalPhermones()$
23:        $BestPath \leftarrow Max(ant.Path, BestPath)$
24:    end for
25: end for
26: return $BestPath$
```

The ant decides which vertex to move to next based on immediate neighbors that have not been visited. Suppose the ant's current vertex is v_i. From the set of adjacent edges $e_{i,j}$, the ant filters out edges it has already visited. If the ant decides to explore, then it randomly selects based on a weighted probability $p_{explore}$ defined as desirability across all options.

$$p_{explore} = \frac{\tau_{i,j} \cdot \eta_{i,j}}{\sum_{0}^{k \in \mathit{NBRS}_i} \tau_{i,k} \cdot \eta_{i,k}} \tag{5.5}$$

If the ant decides to exploit, then it selects the edge with maximum desirability.

$$argmax_{k \in \mathit{NBRS}_i}(\tau_{i,k} \cdot \eta_{i,k}) \tag{5.6}$$

Once the next vertex v_j has been decided, we create a path from v_i along edge $e_{i,j}$. In line 14, the ants apply a local update phermones $applyLocalPhermones$ along edge $e_{i,j}$ to discourage subsequent ants from following to create path diversity. This phermone update uses a learning rate u_{local} to decide how quickly to degrade the phermone. τ_0 is defined to be the total reward collected from the best initial path generated pft_{init} Schilde et al. [2009].

$$(1 - u_{local}) \cdot \tau_{i,j} + u_{local} \cdot \tau_0 \tag{5.7}$$

Once the set of ants has completed their paths, we perform a global phermone update u_{global} before spawning another set of ants. While the local phermone update decreases the attractiveness of edges traversed, the global phermone update increases the attractiveness if the ant identified a path with high rewards. Each ant computes the total reward collected along its path pft_{tot} and then we identify the path resulting in the maximum total reward pft_{best}. Along edges of the best path, we apply the following global phermone update $applyGlobalPhermones$.

$$(1 - u_{global}) \cdot \tau_{i,j} + u_{global} \cdot pft_{best} \tag{5.8}$$

The output of ACO is the overall best path $PathBest$ across all iterations. This path will best utilize the robot's time for the purpose of data collection.

Evaluation

We want to show how ACO better prioritizes where the robot should navigate for sensor measurement needs. For this experiment, we randomly distribute binary rewards across different proportions of regions across the environment. Each approach is given starting location in the center at coordinates $(0, 0)$ and given enough time for the robot to cover 150 grid regions.

We compare several different algorithms. We run ACO with 20 colonies using 10 ants each and both local and global phermone updates set to be 10%. For ACO, we vary the percentage of exploitation between 0% and 100%. 0% creates a colony of ants that always explore based on a weight sum of phermone information. 100% creates a colony of ants that ants always perform exploitation; in essence, following the initial randomly generated path computed without phermone information. The greedy algorithm selects the neighboring vertex with hihgest reward, which makes it a greedy local search algorithm. The random algorithm differs from ACO in that it always randomly selects vertices uniformly at random.

In Figure 5.6, we keeping the data collection time fixed but vary the amount of rewards distributed across the regions. Naturally, all approaches collect more rewards as the amount of rewards to collect increases. Randomized approaches like ACO quickly distinguish themselves from greedy efforts operating off of local reward information. ACO further separates itself from purely random efforts as the amount of rewards increase, showing the benefit of information captured by phermones shared across simulated ants.

Notice that ACO improves after multiple iterations of colony refinement by converging to better paths. This is evidenced by poor performance of ACO with 100% exploitation. The other ACO approaches outperform the purely uniformly random approach, especially as the amount of rewards increases. This suggests that information shared by the phermone map is valuable in finding high reward paths. This makes sense as the random approach generates paths in an undirected fashion while ACO takes advantages of history. These approaches do not converge even when the environment is fully covered by rewards because we are using one-time rewards. When paths end up in situations like dead end hallways, then paths are sometimes forced to pass through already visited vertices that no longer provide rewards. As we can see, ACO is a simple yet effective solution for generating data collection trajectories when given a spatial snapshot of sensor measurement needs.

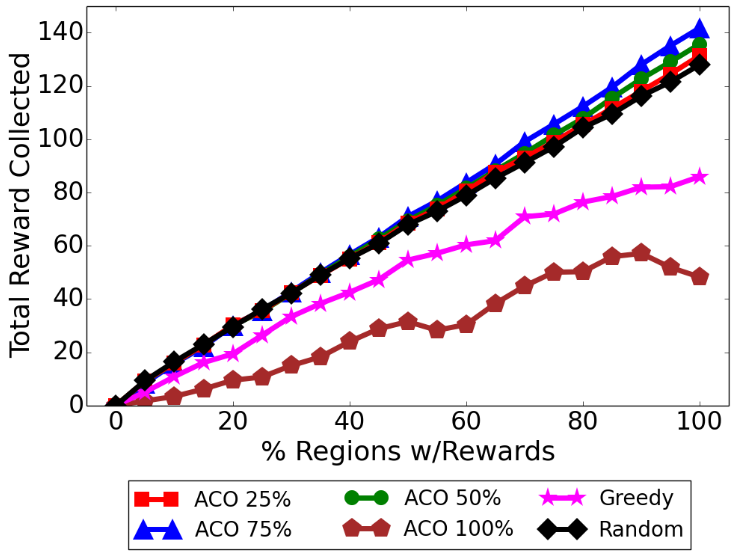


Figure 5.6: Total reward collected as more rewards distributed across the environment.

5.2.2 Trajectories for Temporal Measurement Needs

Measurement value of a particular location may also vary over time (e.g. wireless usage may decrease when classes end). Generating trajectories that incorporate the time value of measurements are traditionally called the orienteering problem with time windows (OPTW) Kantor and Rosenwein [1992]. Our formulation of discrete rewards for each time slot simplifies the search since we know measurement rewards will change together at specific times. We assume that sensor measurements collected in one time slot have no affect on the value of measurements in another time slot. For example, we assume that rewards for measurements at a location within the 11 AM to 1 PM time slot are independent of those within the 1 PM to 3 PM time slot. Therefore, the idea is to independently run a separate trajectory search across each unique reward snapshot with the additional constraint that the starting position of iteration $i + 1$ is the ending position of iteration i. By combining the trajectories into a single one, we effectively have a path that incorporates both spatial and temporal measurement needs.

Evaluation

We want to show the importance of efforts that consider the time value of measurements. If one continues to execute a trajectory based on reward snapshots that no longer apply, then it is inevitable that their trajectories will fail to effectively utilize the robot's limited collection time.

In our experiment, the robot is given enough time to traverse 300 grid regions. We randomly distribute binary rewards across 50 of these grids for each spatial reward snapshot. When there are multiple time slots, we generate independent spatial reward snapshots for each time slot. Efforts that do not consider the time value of measurements compute paths from only the initial reward snapshot to compute a path.

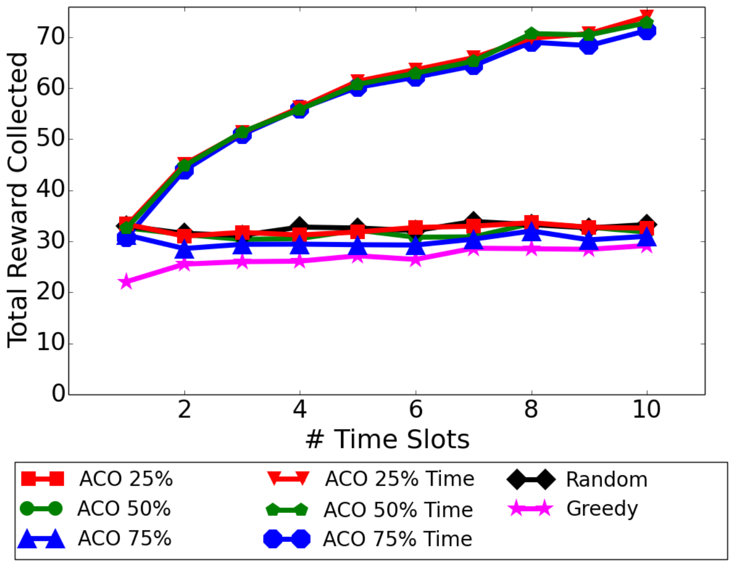


Figure 5.7: Total reward as time value of measurements changes more frequently.

In Figure 5.7, we keep the total data collection time fixed but vary the number of time slots each with their own independent spatial reward snapshots. Efforts that incorporate the time value of measurements quickly distinguish themselves from the others by not relying on stale reward snapshots. As the frequency of time slot changes increase, the opportunity to collect more rewards similarly increases since each spatial reward snapshot is generated independently. When executing paths based on stale reward snapshots, the robot is essentially hoping to stumble upon a high reward locations, which effectively results in strategies no better than random reward collection. In contrast, ACO with time considerations is able to significantly direct its search towards appropriate regions and make the best use of its opportunity.

5.3 Summary

While autonomous robots can collect sensor measurements as they perform user specified tasks, it is desirable for robots to pro-actively navigate to locations of high measurement value when they have spare time. Given a set of measurement priorities across different locations, we presented two approaches for generating pro-active data collection trajectories that differ in how the robot should move across the environment. When we want the robot to move normally along its existing navigation paths, we showed that the discounted reward traveling salesman problem best utilizes the robot's limited data collection time. When we want more fine-grain data collection based on a grid-based representation of the environment, we allow the robot to execute more targeted data collection. We showed that ant colony optimization generates paths that incorporate both the spatial and temporal value of sensor measurements.

Chapter 6

Data Collection with Other Devices: Mobile Phones

There are many different types of mobile platforms and they are not all equally capable as data collectors. A key differentiator is the ability to continuously localize accurately. Naturally, a robot excels at this due to its large form factor that affords powerful sensors, constrained navigation along a 2D plane, and permissive power requirements. Unfortunately, localization is a fundamental challenge for devices like mobile phones that have small form factors, have the freedom to move in many directions, and cannot use power hungry sensors. Despite these challenges, it is desirable to utilize mobile phones for collecting sensor measurements because of their ubiquitous presence across our environments today. In this chapter, we investigate how to effectively recover the locations of sensor measurements captured while mobile devices move across our environments. Instead of relying powerful exteroceptive sensors only found on robots, we focus on using simple motion sensor data that are available on most mobile devices today. In addition, whereas the goal of localization is online location tracking, we focus on the easier problem of offline path identification because sensor data collection does not require continuous, real-time sensor updates.

Our approach is based on the idea that indoor environments are generally fairly constrained (e.g. narrow hallways, small number of intersections). As the device proceeds to move further along, there are fewer possible paths that it could have traversed. For example, if the device moved in a straight line 100 m and there is only one hallway 100 m long, then we can uniquely identify which hallway the device is in but no the direction. If the device proceeds to turn left and this is only possible in one direction along the hallway, then we can uniquely identify where the device must be. Our algorithm essentially takes the device's noisy motion trajectory and performs map matching with a known map of possible navigation trajectories to determine the most likely path the device could have traversed. We first apply this trajectory snapping approach to wheeled robots in order to see if it is possible recover paths traversed from motion trajectories alone and then extended this to mobile phones. Our evaluation shows that trajectory snapping is robust in recovering paths traversed and can even overcome map errors. These results support the idea that less capable mobile platforms like cell phones can be used for collecting sensor measurements across our indoor environments.

6.1 Trajectory Snapping for Robots

Our first step is to see if one can recover the path traversed by a mobile device in an indoor environment using only motion sensor data. We will test the hypothesis that our indoor environments have sufficiently limited navigation trajectories such that we can uniquely identify where the mobile device must have gone. The challenge in relying on motion trajectories is that dead reckoning accumulate significant drift over time, resulting in poor location estimates. The intuition of trajectory snapping is to incrementally force motion trajectory segments to follow the navigation graph that specifies where the device can actually go. The algorithm is inspired by map matching that has been applied successfully for outdoor, vehicular GPS. Our approach applies this technique to an indoor setting where we need to overcome the fundamental differences between globally accurate GPS measurements and locally motion trajectory data. We demonstrate trajectory snapping can robustly recover the paths traversed even when map errors exist.

We divide our algorithm into several pieces. First, we divide the entire motion trajectory into smaller, piecewise linear trajectory segments. Next, we perform map matching by iteratively snapping each trajectory segment to hallways specified by a given navigation graph. This process eliminates unlikely paths based on a known sensor noise model of the motion data. We then consider map errors by reasoning about where the motion trajectory stop and resumed following the given navigation map. Our evaluation shows that trajectory snapping is robust to sensor noise and that it can handle imperfect maps.

6.1.1 Trajectory Segmentation

Trajectory segmentation partitions the motion trajectory into piecewise segments. This effectively separates the trajectory into forward motions and turns so that map matching can map them to edges of the given navigation graph. In this section, we will define relevant background and definitions: the topological map, segmented trajectory, and sensor noise model. We also define dead reckoning to highlight the difficulty of recovering paths traversed in the presence of sensor noise.

Environment Maps

There are different types of map representations. A common map representation is an obstacle map like the one shown in Figure 6.1a that shows where walls in our environment can be found. This map was generated from a 2D blue print where each wall is represented as a line segment. In contrast, a navigation map is a graph showing which routes a device can traverse. In indoor environments, this would be hallways and intersections. In outdoor environments, this would be roads and junctions. The navigation map representation has played a pivotal role in the success of outdoor GPS navigation algorithms so it is natural that our approach uses navigation maps in an indoor setting.

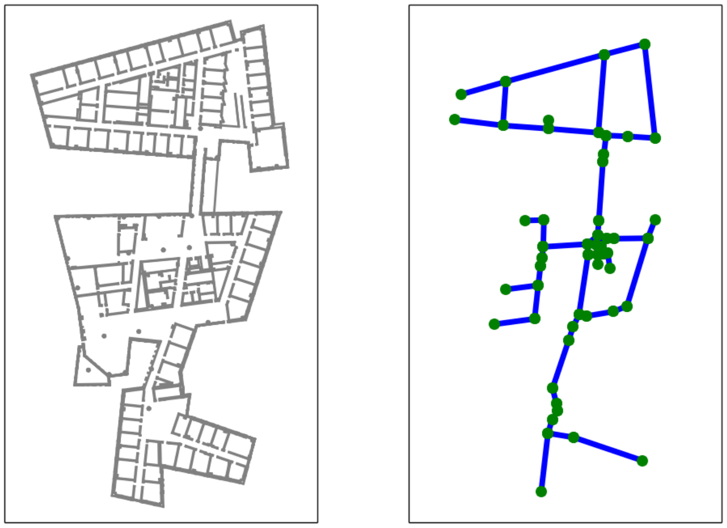

(a) Geometric obstacle map. (b) Navigation map.

Figure 6.1: Examples of obstacle and navigation map of the GHC building.

Let us more concretely define the navigation graph. If we consider an indoor office-like environment, then a node is an intersection and an edge is a hallway, where edges allow transitions between pairs of nodes. Such a representation assumes that the device commits to following a particular topological edge until an intersection point is reached. We will define a node as an intersection point (x, y) in global map coordinates. An edge is a transition between two nodes $((x_1, y_1), (x_2, y_2))$.

Dead Reckoning

There are many possible paths that a device can traverse. Dead reckoning refers to the process of estimating the device's position by integrating its motion data across the entire trajectory. We will see in the definition of dead reckoning how drift errors can accumulate over time due to integration errors that accumulate rapidly.

We will define a device's pose to be a particular position (x, y) and heading h in the coordinates of the topological map (x, y, h). Given an initial pose, dead reckoning will recursively update its pose from the arriving motion updates. Dead reckoning computes its current pose x_i, y_i, h_i using its prior pose $x_{i-1}, y_{i-1}, h_{i-1}$ and current motion update dm_i, dr_i. It computes the global displacement from the motion update and then adds it to the prior pose.

$$\begin{aligned}x_i &= x_{i-1} + sin(h_{i-1} + dr_i) * dm_i \\ y_i &= y_{i-1} + cos(h_{i-1} + dr_i) * dm_i \\ h_i &= h_{i-1} + dr_i\end{aligned}$$

Due to the recursive computation, we can see why errors can accumulate very rapidly, resulting in significant drift over time.

Segmented Trajectory

Given the difficulty of dead reckoning, our approach will first separate the motion trajectory into trajectory segments. We describe our approach to generating trajectory segments from the motion data. Each odometry update u is the change in forward motion dm and rotation dr since the previous update. The trajectory t is then a sequence of motion updates $(u_1, u_2, ...)$ ordered by time. Odometry updates typically occur very frequently (20 Hz for our robot) and capture very small changes in motion as shown in Figure 6.2.

$$\begin{aligned}u &= (dm, dr) \\ t &= u_1, u_2, ...\end{aligned}$$

With such high resolution motion data, each update is insignificant on its own. Therefore, we turn the trajectory into a much smaller set of significant changes in motion U. This assumes that the device's motion can be characterized as a single, significant motion forward dM followed by a rotation dR. This is reasonable in office-like indoor environments where forward motions occurs while traversing a hallway followed by rotations to transition at an intersection. Figure 6.3 shows the segmentation of a trajectory, which appears to already significantly reduce drift. The result is a much shorter segmented trajectory T that will be computationally feasible for our algorithm to recover route candidates.

$$\begin{aligned}U &= (dM, dR) \\ T &= (U_1, U_2, ...)\end{aligned}$$

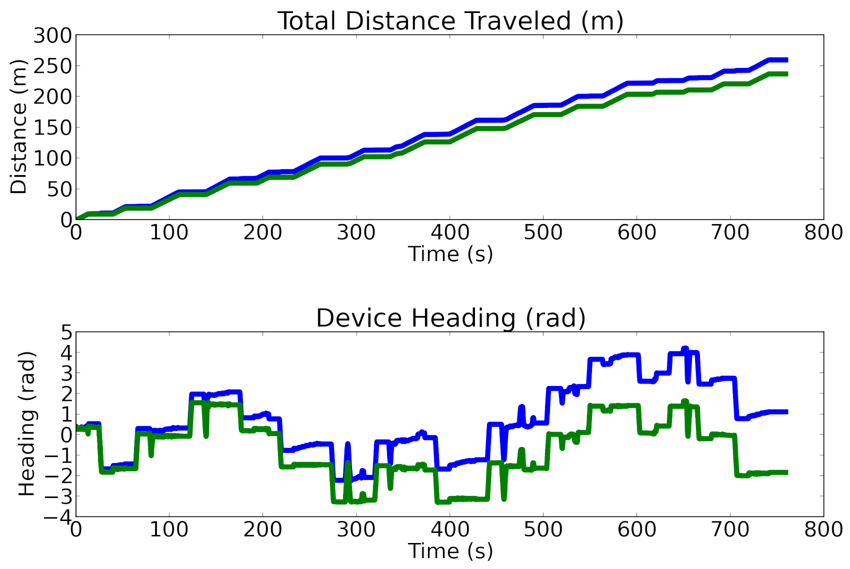


Figure 6.2: Raw motion sensor data (green) compared to the ground truth from LIDAR (blue).

Sensor Noise Model

To generate the sensor noise model, we want to compare the ground truth trajectory to the noisy trajectory from motion data. As shown in Figure 6.3, we can directly see the effect of drift when performing dead reckoning from noisy motion data. Instead of map matching on each of these fine-grain motion update, we will apply it to the segmented trajectory that we can see continues to preserve the shape of the motion trajectory. Let us now define the sensor noise model for the trajectory segments.

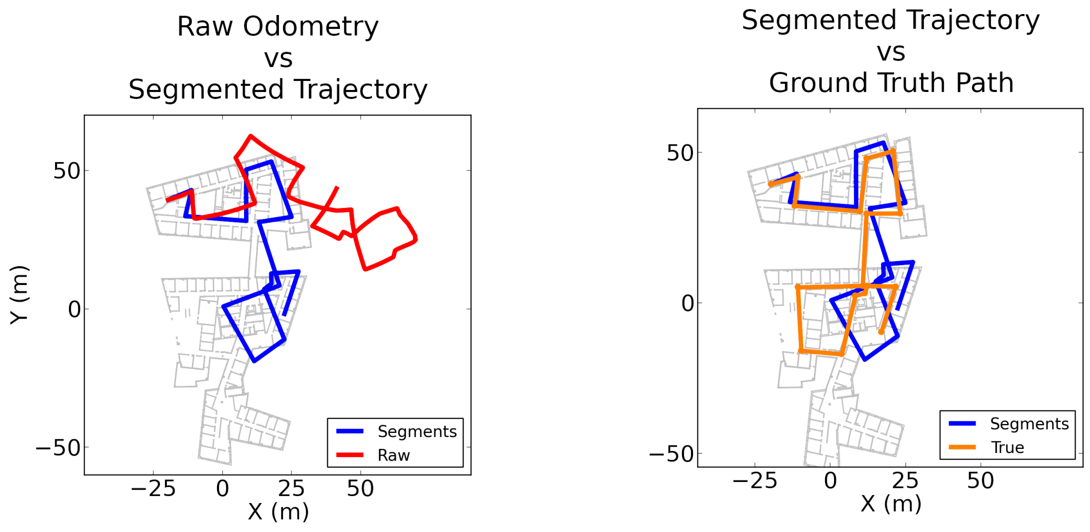


Figure 6.3: Turning raw motion trajectory into trajectory segments for map matching.

We define the error of a single pair of motion updates to be $(\epsilon M_i, \epsilon R_i)$, which is the percentage difference of distance traveled ϵM_i and the difference in rotation ϵR_i.

$$\epsilon M_i = \frac{dM_i^{expect} - dM_i^{actual}}{dM_i^{expect}}$$
$$\epsilon R_i = R_i^{expect} - R_i^{actual}$$

We define the error over the entire route $(\eta M_{route}, \eta R_{route})$ to be maximum over the entire sequence of motion segments. As a result, the noise required to reshape a trajectory is related to the worst behaving snap and not the total sum of noise errors. This will ensure that we can compare errors across routes of different lengths.

The noise model provided to our algorithm will also be the pair $(\eta M_{model}, \eta R_{model})$. Our algorithm searches for all routes that fit within the noise model. As a result, the noise model must account for deviations from both odometry errors and the maximum deviation from the topological map. Its values must bound the worst case snapping errors.

$$\eta M = max(\epsilon M_1, \epsilon M_2, ...)$$
$$\eta R = max(\epsilon R_1, \epsilon R_2, ...)$$

6.1.2 Indoor Map Matching with Navigation Maps

The snapping algorithm assumes that we are given the topological map, sensor noise model, and segmented motion trajectory. The algorithm prunes possible paths that fall outside the given sensor noise model and then ranks the remaining paths based on errors across the entire trajectory. The key contribution is fully utilizing the entire motion trajectory to determine where the device must have gone.

The snapping algorithm essentially searches across all possible paths and snaps them to the motion trajectory segments. A snap refers to matching a trajectory segment to some edge in the navigation graph from some starting state. For example, consider the top left starting location in Figure 6.3 comparing the segmented trajectory versus the ground truth path. We need to make two modifications so that the first trajectory segment snaps to the ground truth edge: a slight clockwise rotation around the starting position and then a minor stretch. Once modified, we are not at a new position and orientation and proceed to repeat the process until every trajectory segment is snapped to a map edge.

Unfortunately, there are often multiple different map edges to snap to so we use a search tree to explore all possible choices that follow the given sensor noise model. Each level in the tree corresponds to the corresponding index of the motion update in the trajectory segment. It is effectively a brute force search where the maximum depth of the search tree is bounded by the number of trajectory segment updates. There are several key reasons why this algorithm is computationally feasible. First, because trajectory segments represent coarse-grain motion updates, we have significantly fewer segments to snap when compared to the total number of fine-grain motion updates captured at 20 Hz. Second, branches can be pruned when a snap falls out of bounds of the given sensor noise model; therefore, devices with more restricted sensor noise have a reduced the search space. Finally, we only consider navigation edge choices that are direct neighbors of the end point of the previously snapped trajectory segment. This ensures that the final path recovered is contiguous and heavily limits the number of map edges to consider (because most intersections in indoor environments only connect to a small number of hallways).

Let us more concretely define the search space with the following rules:

1. A node's value is the device's current pose (x, y, h).
2. A node's pose position (x, y) should correspond to an intersection in the topological map.
3. A parent node at level i can only have child nodes at level $i + 1$.
4. The position of a parent node (x_p, y_p) and child node (x_c, y_c) must have a connecting edge in the topological map.
5. A parent at level i with pose (x_p, y_p, h_p) can have a child node with pose (x_c, y_c, h_c) if applying the motion update (dM_i, dR_i) results in a snapping error $(\epsilon M_i, \epsilon R_i)$ within the bounds of the given noise model $(\eta M_{model}, \eta R_{model})$

The algorithm begins by creating a set of root nodes at level 0 initialized to one of the intersection points in the topological map. For each node, we then recursively compute the reachable intersections in the topological map from the node's current pose. A child node is created if the error in the expected and actual motion updates are within the sensor noise model $(\eta M_{model}, \eta R_{model})$. If the search reaches a depth where there are no more motion updates, then all trajectory segments have been successfully snapped. Once the search completes, we have a set of successfully snapped routes from which we can select the optimal.

Too restrictive of a noise model may not successfully recover the actual route. A more permissive sensor noise model will naturally increase the size of the search space; however, it does not reduce the quality of the routes discovered. In fact, it may discover many alternate routes, which is not necessarily undesirable because we know about the error bounds $(\eta M, \eta R)$ for each route. If there is one route that has a snapping error significantly lower than all others, then it is very likely that it is the actual route. In contrast, if several routes have similar errors, then the best route is now more ambiguous. The strength of our algorithm is the ability to effectively quantify confidence in the recovered routes.

6.1.3 Overcoming Navigation Map Errors

A limitation of our snapping algorithm is the assumption of a perfect navigation map. Unfortunately, it is inevitable that errors will be present in the map so it would be desirable for our algorithm to gracefully handle these situations.

For example, a missing navigation edge has two possible outcomes: either we recover an incorrect path or fail to recover a path. One way to reduce missing navigation edges is to automatically connect all intersection points with visibility to one another. An example is shown in Figure 6.4, where open areas are densely covered with numerous edges. In exchange for a larger search space, we mitigate the possibility of missing navigation edges.

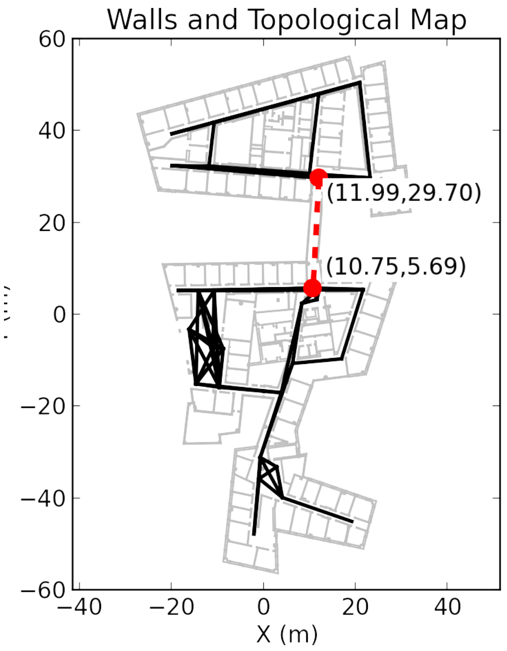


Figure 6.4: Example of navigation graph with unmarked edge.

Unfortunately, it is still possible to have missing navigation edges. For example, suppose we are missing the hallway edge shown in Figure 6.4. One approach to is to soften the hard constraint that a parent and its child must be connected at their ending and starting position, respectively. This property ensure contiguous paths but means that if an edge is missing, we cannot recover the path. The solution is to set a user specified limit on the number of nodes where this property does not need to hold.

This simple extension allows our algorithm to be flexible enough to address a few missing navigation edges while still taking advantage of connectivity in the topological map to find only the most likely routes and also remaining computationally feasible.

6.1.4 Evaluation: Trajectory Snapping Robot Motion

In our evaluation, we record odometry and localization data from an autonomous, omni-directional wheeled robot Veloso et al. [2012a] equipped with odometry, LIDAR, and Kinect sensors. The robot is directed to visit a sequence of waypoints throughout the environment. Each waypoint corresponds to a node on the topological map. The recorded odometry trajectory segments are given to our algorithm to see if it can recover the route.

This setup allows us to evaluate our algorithm on repeatable, real odometry sensor data for which we have the ground truth. It also introduces additional challenges for our algorithm because the robot's autonomous behavior in avoiding obstacles or wall following may violate the assumptions of direct motion between waypoints. We will show that our algorithm can still successfully recover all routes in the presence of such challenges.

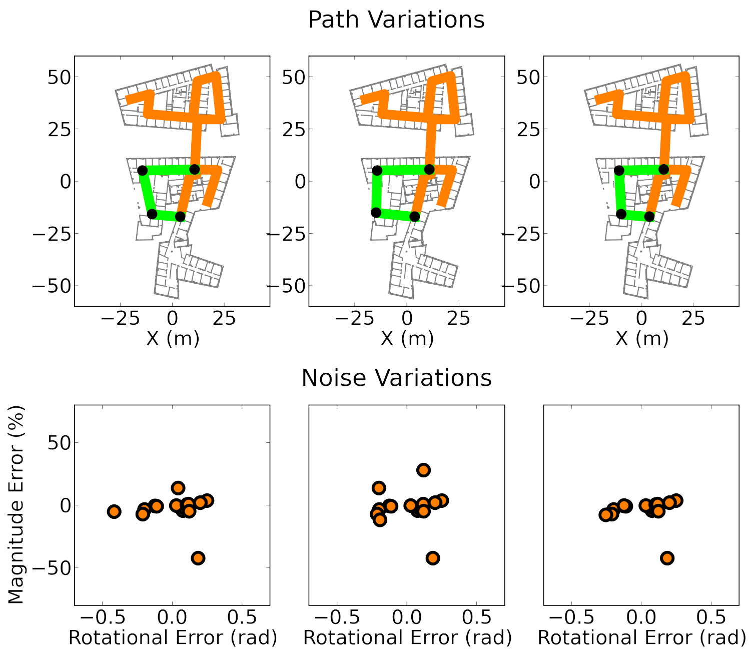


Figure 6.5: Possible routes and their corresponding snapping modifications.

Recovered Routes With the given sequence of trajectory segments from the robot's odometry, our algorithm returns a set of all recovered routes. Figure 6.5 shows a set of routes recovered from the trajectory segments from Figure 6.3. Notice that the recovered routes shown looks remarkably similar except for a select few highlighted segments, which are all located in an open area where there were several possible choices. This similarity means we can be fairly confident that we have correctly identified most of the actual route. If the algorithm would have found routes with completely different start and end positions, then it would be more difficult to be confident about choosing a particular route. These types of conclusions are made possible only because our algorithm is able to provide us with the set of all candidate routes for the particular trajectory and map. Such conclusions would not be possible with other approaches including particle filters that must work with a much larger search space.

Optimal Routes The robot was directed to follow the sequence of waypoints specified by the actual routes shown in Figure 6.6. Each route was repeated for five iterations to evaluate repeatability of our algorithm. Without any corrections, the recorded trajectory segments result

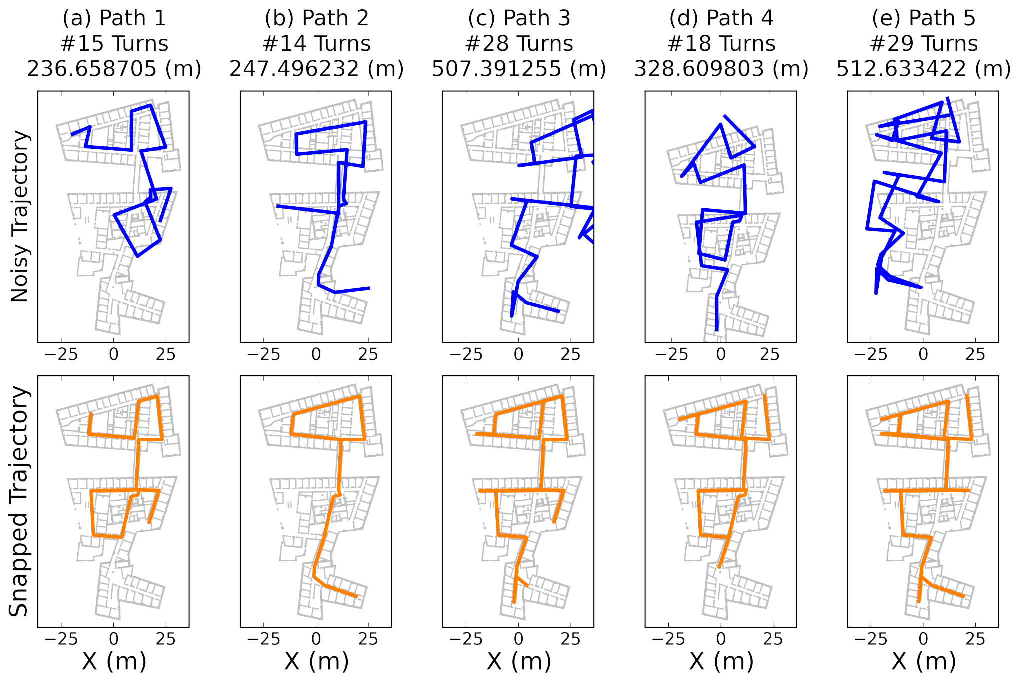


Figure 6.6: Example routes that trajectory snapping is able to recover.

in fairly poor routes with many being completely outside the map. Drift errors for the routes differ depending on a combination of difficulty and length of the route, directions of turns, and other autonomous behaviors.

Iter	Route				
	1	2	3	4	5
1	**2** (6)	**3** (6)	**1** (24)	**1** (9)	**1** (96)
2	**1** (6)	**3** (6)	**2** (24)	**1** (18)	**1** (96)
3	**1** (6)	**3** (6)	**1** (24)	**1** (9)	**1** (96)
4	**1** (6)	**3** (6)	**1** (24)	**1** (18)	**1** (96)
5	**1** (6)	**3** (6)	**2** (24)	**4** (18)	**1** (96)

Figure 6.7: Rank of ground truth path among feasible routes.

Each route was repeated for five iterations. The algorithm was given a very permissive noise model of $(eM, eR) = (65\%, .8radians)$. From the set of recovered routes, we rank them based on a simple sum of squared distance on the maximum snapping error of each route $\sqrt{eM^2 + eR^2}$. Figure 6.7 shows the ranking of the actual route to the total number of recovered routes. Ranking is computed with a sum of squared difference of snapping errors. In most cases, the actual route has the highest rank. As shown in Figure 6.5, while the number of alternative routes appears large, many of them are small variations of the overall actual route. Nevertheless, the success of our simple ranking algorithm affirms the observation that the actual route will typically have the most conservative snapping error bounds.

6.2 Trajectory Snapping for Mobile Phones

Extending trajectory snapping to mobile phones is challenging because phones have more degrees of freedom and lack powerful exteroceptive sensors found on robots so we need to make some assumptions. We assume that users continuously move across the environment and carry their cell phones under consistent conditions (e.g. phone flat against shirt pocket or held in hand) and continuously move across the environment. By carefully extracting motion data in the form of steps and heading, we are able to recover the underlying path traversed by the user under different phone orientations.

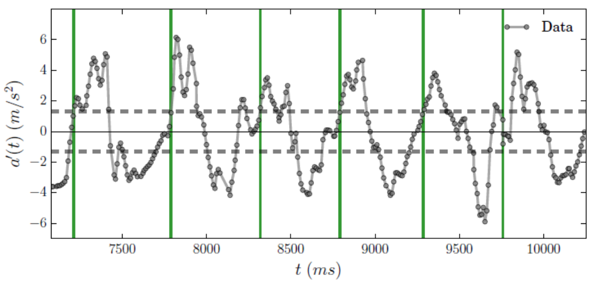
The two major challenges in extending trajectory snapping from robots to mobile phones are robust extraction of trajectory segments from cell phone motion data and ensuring that trajectory snapping continues to be robust when using even noisier motion data. In this section, we focus on extracting trajectory segments by focusing on identifying steps and heading based on mobile phone motion data. This is challenge because user activity, device type, and phone position on the body all contribute to the difficulty of robustly extracting motion trajectories. We discuss how we are able to extract both steps and heading of the mobile device as long as the cell phone is held in a fixed orientation relative to the walker (i.e. in the hand or in a pocket). Our evaluation shows that trajectory snapping continues to be robust despite the more permissive sensor noise model that mobile devices are subject to.

6.2.1 Step Extraction

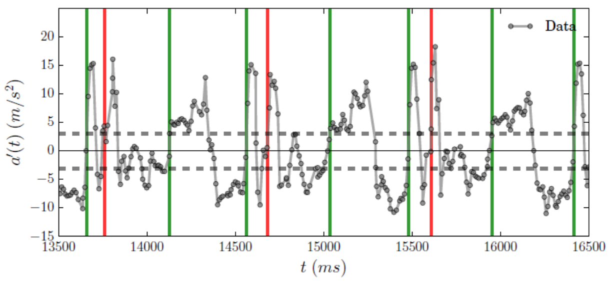
Instead of directly using the 3-dimensional acceleration signal, we use the magnitude of acceleration normalized to zero-mean. Thus, if $a(t) = (a_x(t), a_y(t), a_z(t))$ denotes the acceleration signal of the phone at time t, we consider the quantity $a'(t) = |a(t)| - \overline{|a(t)|}$. We denote the corresponding array of normalized acceleration measurements $\{a'(t_1), \ldots, a'(t_n)\}$ by $\mathcal{A}$.

Naive Threshold Crossings The most naive approach would identify steps as zero-crossings of the normalized acceleration; unfortunately, this is unreliable in practice due to noisy sensor measurements. A slightly better approach applies a lower and upper threshold that the normalized accelerations must both cross in order to register as a step. This method is efficient and does not require advance knowledge about the period of each step; however, it assumes that the step signal is sinusoidal. It will not be able to cope with more complex accelerometer signals.

An example is shown in Figure 6.8 where lower and upper thresholds are shown with horizontal, dashed grey lines. The vertical lines indicates the estimate of where each step begins, where green indicates correct registrations and red incorrect steps. For example, Figure 6.8a shows how the threshold crossing approach is fairly accurate when the phone is held in the hand. In contrast, Figure 6.8b shows how such an approach fails when the device is placed in the pocket, resulting in over-counting of steps.



(a) Correct



(b) Incorrect

Figure 6.8: Counting steps with naive threshold crossing.

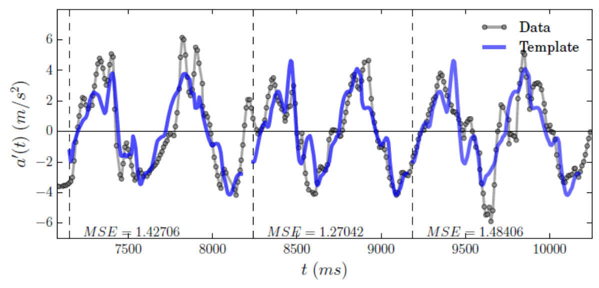
Template Matching for Steps With template matching, we assume a periodic function between acceleration and time but we do not manually design conditions based on prior assumptions about the shape of the acceleration signal. Instead, our template-matching algorithm automatically learns the periodic function and identifies steps based on a learned template.

There are three major steps: estimating the periodicity of steps, identifying the initial template, and then counting the number of repetitions of the template. Periodicity arises in the normalized acceleration signal as the human takes turns between stepping with their left and right foot (so each template identifies two steps). We apply a Fast Fourier Transform (FFT) to the signal to extract the period P of these two steps.

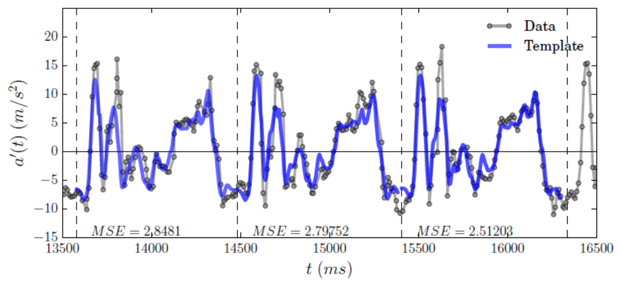
To identify an initial step template, we look at an initial time interval $I_0 = [t_0, t_0 + P]$, where t_0 is the initial time when the human started walking. Within interval I_0, we find time $t_m \in I_0$ that maximizes the acceleration signal because is usually corresponds to the person's foot hitting the ground. Roughly estimating that each step takes the same amount of time, we then choose the time interval $[t_m - \frac{P}{4}, t_m + \frac{3P}{4}]$ to represent the step template.

Finally, we count steps across the entire normalized acceleration signal by finding step template matches. For each step i, we search for the time interval $I_i = [t_i, t_i + P]$ during which the step occurred. While human steps are not perfectly synchronized, we expect that the starting time for step i will quickly follow the end of the previous step, $\{t_{i-1} + P < t_i < t_{i-1} + 5P/4\}$. We optimize for the best starting time t_i by minimizing the mean squared error of the template match. This process is repeated for each step until the entire normalized acceleration signal is processed.

Figure 6.9 shows the result of applying template matching to the same readings from Figure 6.8. Template intervals are indicated by dashed grey lines and the template is overlayed in blue. Nevertheless, these results indicate that template matching can tolerate many different shapes of steps as long as the acceleration data is periodic. In addition, it can adjust to variations over time including slight stops and changes in pace. This explains the minor gaps between several of the template as the algorithm adapts to imperfectly synchronized steps.



(a) Correct



(b) Incorrect

Figure 6.9: Counting steps by template matching.

6.2.2 Identifying Turns

We also wish to extract changes in heading regardless of phone orientation. Compass readings are useful only if the relative orientation of the phone and body are known for all measurements. Additionally, compass errors may result from local magnetic fields or device misalignment. According to Gusenbauer et al. [2010], typical mobile device compass errors, when compared to known headings, are at least 5 degrees. We therefore use both gyroscope and accelerometer data to get a raw estimate of the headings.

Assuming that the main component of acceleration is gravity (true in our context), the direction of acceleration can be used to determine where the ground is, no matter the orientation of the phone. We then approximate the angular velocity of the turning of the user's body using the acceleration-direction component of the gyroscope reading. The following equation demonstrates the relationship between ω, the estimated angular velocity of the turning of the user, and the gyroscope and accelerometer data:

$$\omega = \frac{\omega_x a_x + \omega_y a_y + \omega_z a_z}{\sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}},$$

where ω_x, ω_y and ω_z are the measures of angular velocity in the three axes of the phone.

The angular velocity ω is a good measure of the turning of the device carrier if the phone is not shaking severely relative to the carrier's body. No knowledge about how the phone is placed is required so the user may change the phone's position throughout the data collection period (e.g. taking the phone from the pocket and holding it in the hand). However, in our experiments this method will still perform poorly if the user swings their arms significantly or places the phone in a very loose pocket.

6.2.3 Segment Identification for Cell Phones

We regard a series of consecutive steps with small total change in heading as a straight walk and a series with larger change of heading as a turn. Explicitly, consider a sequence of steps $\{x_i\}_{i=1}^m$ with respective headings $\{\psi_i\}_{i=1}^m$, let $\tau > 0$ denote the turning threshold, and w, an odd integer greater than 3, the window length parameter. For each i, consider the window W_i of w consecutive steps centered at step i (restricting the window size for i near 1 and m as necessary), and let $d\psi_i$ be the largest difference between all headings in W_i. Given a consecutive sequence $d\psi_i, \ldots, d\psi_{i+k}$, $k \geq 0$, each of which is greater than τ, we say a turn occurred at index $median(\{i, \ldots, i+k\})$. After the turns are identified, the segments are calculated as the summed length of the steps between two turns and the angles of the segments are calculated as difference in headings between consecutive turns.

Regarding the choice of parameters, w should be larger when the space is more open. For example, in a shopping mall with wide intersections, people may take more steps to finish a turn than in office environments. However, with a large w the window sometimes can contain more than one turn, resulting in mis-identification of turns. On the other hand, the turning threshold τ should be small enough to recognize turns, but much larger than the error in individual step headings, so avoid over-counting turns.

Note that the task of separating turns and straight walks can be ambiguous due to the difficulty in defining a turn. One can walk in a curve along a straight hallway and in a straight line along a shallow-angled intersection. This kind of ambiguity can lead to error in segment identification, but a flexible topological map can account for this. In general, we would rather consider two real segments as one rather than break one actual segment into two. For example, in Figure 6.10, the segment identification algorithm ignored two small turns. The modification of the topological map is to actually allow some intersections that pass through walls to be connected by artificial "hallways". To build this topological map, one can first connect all adjacent intersections without passing through walls, and then artificially connect non-adjacent (but close) intersections which don't require a big turn to reach one from the other.

6.2.4 Evaluation

We implement the snapping algorithm in NYU's Center for Urban Science and Progress, which occupies the 19th floor of 1 MetroTech Center in downtown Brooklyn, NY. This setting is the floor of an indoor office building, with narrow corridors, offices, and cubicles. Most hallways are between 5 and 50 meters long, and most angles between corridors are right angles. Figure 6.10 shows both the underlying map of CUSP, along with the actual 12-segment long path traversed, as well as the topological map used for snapping.

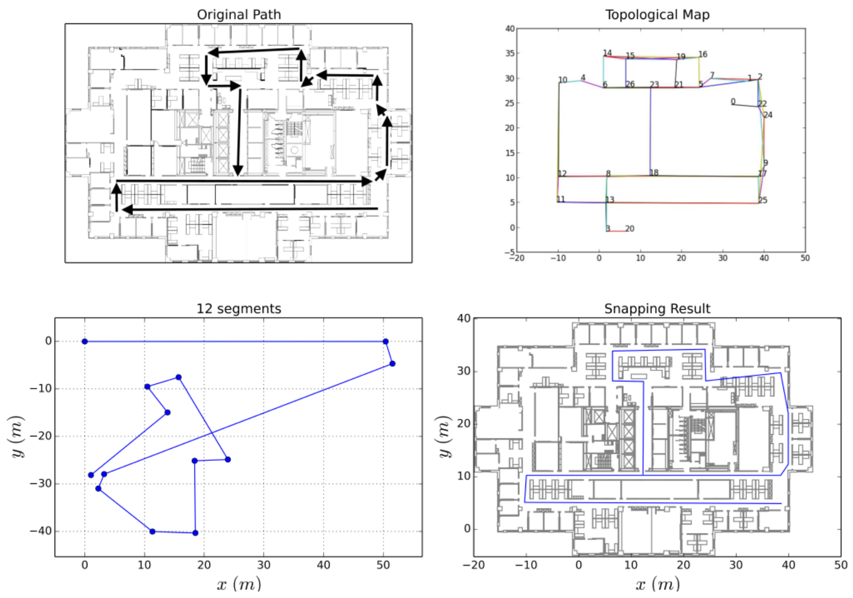


Figure 6.10: Trajectory from real cell phone motion data at NYU.

	Γ_2	Γ_3	Γ_4	Γ_5	Γ_6	Γ_7	Γ_8	Γ_9	Γ_{10}	Γ_{11}	Γ_{12}
Segment 1	2	2	1	1	1	1	1	1	1	1	1
Segment 2	2	2	1	1	1	1	1	1	1	1	1
Segment 3	-	2	1	1	1	1	1	1	1	1	1
Segment 4	-	-	1	1	1	1	1	1	1	1	1
Segment 5	-	-	-	2	2	1	1	1	1	1	1
Segment 6	-	-	-	-	3	1	1	1	1	1	1
Segment 7	-	-	-	-	-	2	1	1	1	1	1
Segment 8	-	-	-	-	-	-	1	1	1	1	1
Segment 9	-	-	-	-	-	-	-	2	2	2	2
Segment 10	-	-	-	-	-	-	-	-	2	2	2
Segment 11	-	-	-	-	-	-	-	-	-	3	2
Segment 12	-	-	-	-	-	-	-	-	-	-	1
Full path	2	2	1	2	3	2	1	2	2	3	2

Data was collected from an Android application installed on the second author's handheld Samsung Galaxy S4 smartphone and converted into a segmented motion trajectory that also appears in Figure 6.10. We used $dm_1 = 0.25$, $dm_2 = 5$, and $da = 0.8$, roughly $\frac{\pi}{4}$ radians, as error thresholds. We took the window length parameter $w = 5$ and the turning threshold $\tau = 0.4$ radians. We also assumed a length of 0.8 meters for every two user steps when converting the raw input data into a segmented trajectory.

We have plotted the result of the snapping algorithm in the lower right panel of Figure 6.10, which conforms quite closely to the actual traversed path, although direction of motion is not indicated. The snapping algorithm correctly snapped all twelve segments to their best match. In this implementation there were two output paths that were admissible within the given error thresholds, but given multiple admissible paths, we rank them by total percentage error (the sum of percent angular and percent length errors over all trajectory segments) and the correct path is selected when choosing the lowest-error path.

Next, we examine how uncertainty in the snapped path varies with the addition of new segments. In particular, we look at both how the total number of possible snapped paths varies, as well as how the number of possible matches for a given segment varies.

First, we introduce some notation to facilitate a discussion of uncertainty in the snapped trajectory as the number of segments in a path increases. For a topological map M, a set of error thresholds E, and an input trajectory of n segments, $\Gamma_n = \sum_{k=1}^{n}(S_k, \theta_k)$ as above. We define

- $\rho_j(S_k) =$ the number of matches for segment S_k, considered as a segment in Γ_j, where $j \geq k$, when Γ_j is snapped to M with thresholds E.
- $\rho(\Gamma_j) =$ the number of trajectories for Γ_j that snap to M with thresholds E.

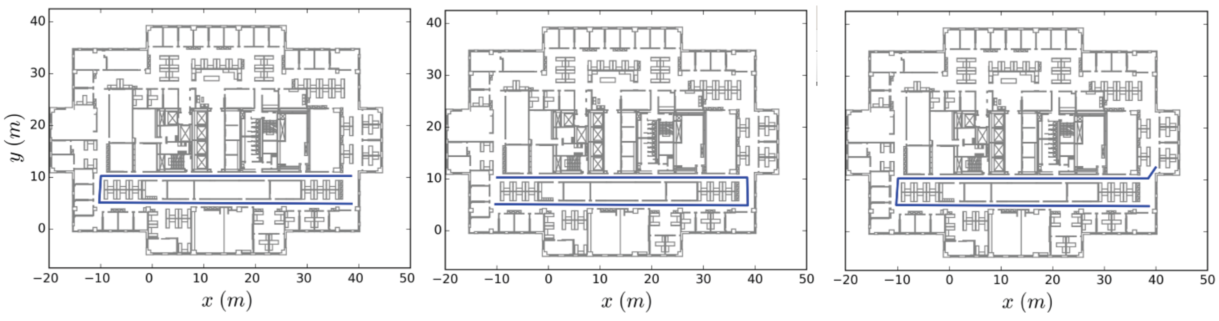


Figure 6.11: Possible paths after snapping a short, three segment partial path.

We expect $\rho_j(S_k)$ to monotonically not increase as j increases, for a fixed k. That is, adding additional constraints to Γ_j can only decrease the number of possible matches for S_k. Intuitively, segments early in a trajectory become "locked in" as newer segments are added, although the overall path may retain ambiguity. We observe this behavior also by looking at the number of matches for segment S_k, $k = 1, \ldots, 12$ in the context of the path traversed at CUSP.

This is displayed in the table in Figure 6.11, where the number of matches for any given segment S_k only stays the same or decreases as the number of segments in the path it belongs to increases. The (i, j) entry in the table above shows the number of possibilities for Segment i

in the partial path Γ_j, where $1 \leq i \leq 12$ and $2 \leq j \leq 12$. The j^{th} entry in the last row in the table gives the total number of possible matches for Γ_j, i.e. the maximum of all entries in the j^{th} column of the table. Note that each of the first twelve rows is monotonically non-increasing as j increases, illustrating that earlier segments are "locked in" as new segments are added to the path. The three plots give an illustration of this for Γ_3 and Γ_4. Specifically, the two left plots demonstrate the two possible matches for Γ_3, and the two possibilities for Segments 1, 2, and 3 in Γ_3. The right plot demonstrates that the addition of Segment 4 gives a unique result for Γ_4, and hence there is now only one possible match for Segments 1, 2, and 3 in Γ_4. Explicitly, S_3, for example, has two possible matches in Γ_2 and Γ_3, then has only one possible match in $\Gamma_4, \ldots, \Gamma_{12}$, as illustrated in the plots in Figure 6.11.

On the other hand, $\rho(\Gamma_j)$ may fluctuate non-monotonically as j increases from 1 to n. For example, $\rho(\Gamma_j) < \rho(\Gamma_{j+1})$ may occur if S_{j+1} matches several segments in the topological map within given error thresholds. On the other hand, $\rho(\Gamma_j) > \rho(\Gamma_{j+1})$ if the addition of S_{j+1} eliminates possible matching trajectories for Γ_j. We observe this behavior in the 12-segment path traversed at CUSP, in the last row of the table. The total number of matches varies between 1 and 3, and increases and decreases as more segments are added.

6.3 Summary

We wanted to use alternative mobile devices like cell phones for collecting sensor measurements that are less capable than autonomous robots but much more ubiquitous. We addressed one of the biggest challenges in collecting meaningful measurements with these devices, which is the difficulty of robust recovery of paths traversed. We introduced a trajectory snapping approach based on outdoor GPS map matching that takes advantage of the limited navigation options due to the structure of indoor environments. We showed this approach works robustly for robot motion data and even overcomes navigation maps with errors. We extended these efforts to motion data from mobile phones and showed that trajectory snapping successfully recovers the paths traversed even when the phone is placed in many different configurations. These efforts suggest that we should be able to utilize a wide range of mobile platforms to help collect sensor measurements about our indoor environments.

Chapter 7

Use of Sensor Maps: Mobile Wireless Handoffs

We have now reached the point where we can ensure up-to-date sensor maps due to a combination of data collection (Chapter 3 and Chapter 5) and data analysis (Chapter 4) efforts. In this chapter, we want to show that fine-grain sensor maps can be valuable for algorithms that take advantage of environment-specific awareness of surrounding conditions. For example, applications include efficient HVAC control from awareness of building occupancy Agarwal et al. [2010], Pérez-Lombard et al. [2008], climate control for preservation of museum pieces Michalski [2007], and effective wireless network management from knowledge about wireless dead spots Mahajan et al. [2006]. Such environment-specific algorithms are in limited use today because it is challenging to ensure up-to-date and fine-grain sensor maps. Our belief is that there remains many opportunities to develop context-aware algorithms. We present a concrete example where wireless maps helped to resolve our autonomous robot's own intermittent wireless connectivity issues. This type of wireless usage pushes the fundamental limitations of existing wireless handoff algorithms, resulting in poor wireless experiences. We introduce map-based wireless handoff algorithms that overcome these challenges by simply using fine-grain awareness about surrounding wireless conditions. We hope this chapter demonstrates the exciting opportunity to transform traditionally difficult problems into more manageable ones via data-driven algorithms that effectively utilize fine-grain sensor maps.

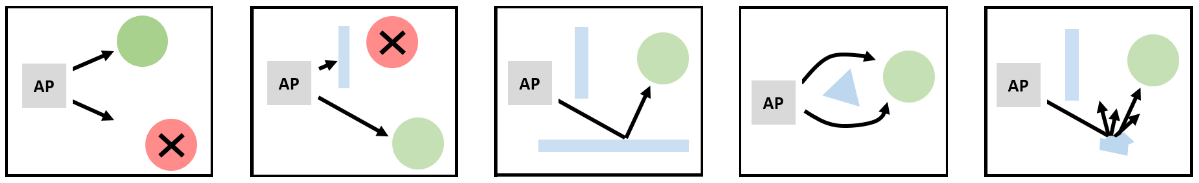
The two primary contributions of this chapter are diagnosis of problems that arise with motion by tracking sensor variations and map-based algorithms that utilize fine-grain sensor maps. We specifically use wireless connectivity as an example to show how a mobile platform enables us to determine the root cause of our wireless connectivity issues and then produce a wireless solution that does not require modifying other wireless devices. This chapter is structured as follows. First, we discuss why wireless conditions are difficult to predict across real environments in order to motivate the need for fine-grain wireless data collection. Next, wireless connectivity issues that arise intermittently while moving can arise from a variety issues (e.g. wireless dead spots, device mis-configuration). By searching for reoccurring patterns that emerged from tracking wireless performance while moving, we are able to identify the root cause of intermittent wireless connectivity. We then develop map-based wireless handoff algorithms that use fine-grain wireless maps to determine which AP to switch to. Unlike many alternate solutions, our approach does not require that we overhaul existing wireless infrastructure or protocol. Nevertheless, our evaluation demonstrates that our handoff algorithm significantly improves actual wireless performance in practice. This is just one example showing the potential practical implications of data-driven algorithms that we hope encourages future efforts to effectively utilize fine-grain sensor maps.

7.1 Challenges of Understanding Wireless Connectivity

In this section, we motivate the need for collecting wireless maps and highlight the difficulty of understanding wireless performance especially when moving. We showed in Chapter 4 that wireless conditions vary wildly across different locations because wireless signal propagation depends on where the wireless device was located. First, we further explain why wireless signals are difficult to predict in practice. As a result, it is more reliable to directly gather wireless measurements like we show in Chapter 3. High RSSI is only a pre-requisite for wireless performance so we introduce other important factors that influence wireless connectivity. This includes wireless channels that enable simultaneous wireless transmissions without interference; the finite wireless spectrum means that it is important to utilize the limited number of wireless channels effectively. In addition, while throughput is a much better reflection of actual data transmission, it depends on the proper operation of multiple network layers. As a result, it is helpful to separately consider RSSI and throughput measurements and isolate the layer in which connectivity failures arise.

Wireless Signal Propagation

Wireless signals have limited range because their energy decays as a function of distance. Many efforts have modeled the signal propagation of WiFi signals when there is clear line of sight and found the received power to decay logarithmically with increasing distance from the transmission source Iskander and Yun [2002], Bahl and Padmanabhan [2000]. Realistic environments introduce additional effects on wireless signals including reflection, diffraction, and scattering as shown in Figure 7.1. There are also small-scale fading effects where the signal strength can vary several magnitudes by just moving fractions of a wavelength. In realistic environments, these factors are difficult to directly measure and their combined effects are difficult to accurately predict because they are so dependent on static and dynamic factors in the surrounding environment Andersen et al. [1995]. Illustrative examples are shown in Figure 7.1 where APs are represented with squares and WiFi clients with circles. WiFi clients marked with an X reflect devices that cannot communicate with the AP.



(a) Attenuation over distance (b) Attenuation by objects (c) Reflection at an angle (d) Diffraction around an object (e) Scattering in many directions

Figure 7.1: Illustrating the various environmental effects on WiFi signal propagation.

Bands and Channels

Recall that 802.11 wireless signals are transmitted over the 2.4 Ghz and 5 Ghz bands that are further subdivided into 12 and 23 channels, respectively. Each channel is centered around a specific frequency. Multiple WiFi channels allows us to perform frequency-division multiplexing, where simultaneous WiFi transmissions on different non-overlapping channels can occur without affecting WiFi performance. There are 23 non-overlapping channels for 5 Ghz but only 3 for 2.4 Ghz (channels 1, 6, and 11). Since many older devices only support 2.4 Ghz, the 2.4 Ghz band is a scarce resource that needs to be shared efficiently. As a result, wireless APs are often re-configured intermittently to adapt to changing wireless conditions and usage patterns.

When multiple WiFi devices transmit simultaneously on the same channel, the transmitted wireless signals may interfere with one another and cause the transmisions to be undecipherable by all receivers, which effectively wastes an opportunity for transmitting wireless data. This type of WiFi interference can significantly degrade WiFi performance. A natural solution is for each WiFi device to transmit at a different time. In 802.11 WiFi, there is no centralized coordination to allocate transmission time like there is for cellular. Instead, 802.11 WiFi uses carrier sense multiple access with collision avoidance (CSMA/CD), where each WiFi device transmits WiFi data when it senses that the channel is idle and waits when it senses other transmissions. 802.11 employs exponential backoffs, where the WiFi device will wait a multiplicatively increasing amount of time each time it tries to transmit but detects other transmissions.

As the number of WiFi devices using a single channel increases, so does the likelihood of interference. Increased interference leads to longer wait times before WiFi devices attempt to transmit. This results in poor WiFi performance and inefficient usage of the WiFi medium. A common example of poor WiFi due to interference is apartment complexes with uncoordinated 2.4 Ghz WiFi routers all operating on the limited set of channels over the same areas resulting in poor WiFi connectivity for all users. Therefore, enterprise management of the WiFi medium is important to reduce erroneous sources of interference and ensure proper allocation of WiFi channels for encouraging more efficient usage of the wireless medium Ramachandran et al. [2006].

Throughput

As shown earlier in Figure 4.8, higher RSSI does not guarantee high throughput but is a prerequisite for strong wireless performance. Throughput defined as the useful number of megabits successfully transferred per second. Overheads like packet headers and other negotiation packets are not included in these measurements. This contrasts bandwidth, which usually reflects the theoretical maximum rate of data transfer. Throughput is a practical although imperfect estimate of the actual quality of a link. In this section, we measure throughput with the iperf tool over TCP. We recognize that TCP is not a perfect indicator of link capacity because it depends on the proper operation of multiple network layers. For example, wireless connectivity requires the wireless device to associate with a nearby AP and ensure the AP is properly configured to forward the device's packet to the wired backbone. On top of this, flow control algorithms at the TCP layer requires some time to ramp up to actual link speeds. Therefore, throughput measurements are affected by multiple factors so many of our results will try to isolate where connectivity failures stem from.

7.2 Diagnosing Connectivity Issues with a Mobile Robot

Wireless connectivity issues could arise from a variety of reasons: poor AP coverage, bad wireless handoffs, interference from surrounding devices, etc. Diagnosing the wireless problems for a specific wireless device in practice is difficult because they depend on environment-specific factors like structure, materials, users, wireless devices, etc. Mobile robots can be a pragmatic solution in searching for the root cause by replicating problematic wireless situations with autonomous navigation and also revealing wireless performance variations. In this section, we first provide relevant background on wireless handoffs and then show how we identified poor wireless handoffs as the cause of the robot's intermittent wireless connectivity.

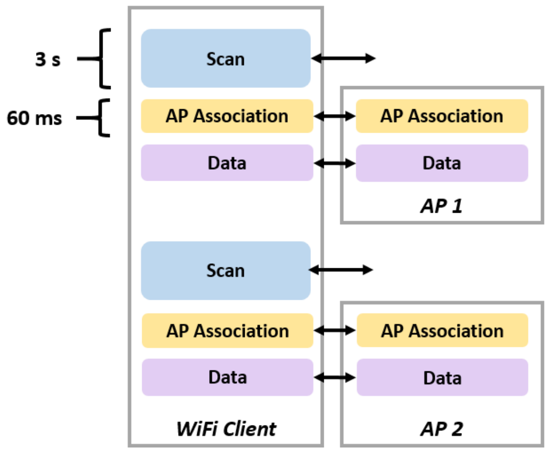
7.2.1 Importance of Wireless Handoffs

Wireless usage can generally be characterized as portability or mobility. Portability is the use of wireless connectivity while stationary at a small number of different locations. Strong wireless performance requires that there are no wireless dead spots at these small sets of locations. Mobility is wireless usage while moving across the environment. Real-time applications that require continuous wireless updates push the boundaries of wireless connectivity because there cannot be any dead spots along the entire path. They also require seamless wireless handoffs as the device moves in and out of range of different wireless access points. We first describe 802.11 WiFi handoff algorithms used in nearly all mobile devices today. We then introduce related research efforts to improve the performance of these algorithms.

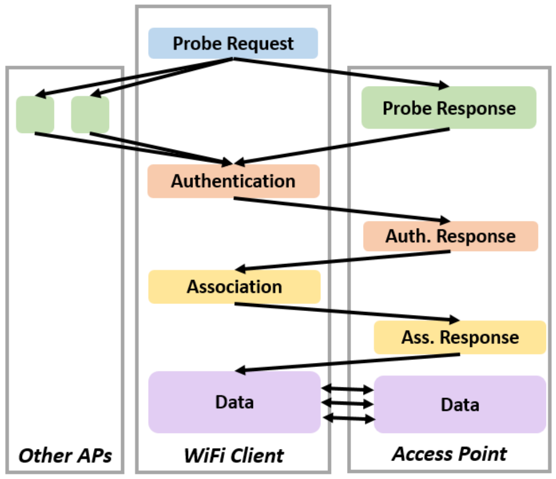
802.11 WiFi Handoff Process

When a wireless client wants to connect to the wireless network, it needs to first identify nearby APs that it can connect to. This is often performed by a scan where the WiFi device broadcasts probe requests and then accumulates responses from nearby APs. As illustrated in Figure 7.2a, a scan often takes 3 to 5 seconds. From the scan results, the wireless clients will identify and connect with the best AP available. The connection is established by exchanging authentication and association frames with the selected AP, which usually takes around 60 ms as shown in Figure 7.2b. If successful, the wireless client will be associated with the AP and can proceed to exchange data across the network. If the wireless client should move out of range of its current AP, it will need to disassociate with its current AP and repeat the process of finding and associating to another AP.

Notice that disassociation occurs before a scan, which means the WiFi device has no connectivity during a scan. This is due to the inability of WiFi devices to simultaneously iterate through all WiFi channels while also communicate with its associated AP on a separate WiFi channel. These switches between APs are called AP handoffs.



(a) 802.11 Handoff Timings



(b) 802.11 Handoff Protocol

Figure 7.2: 802.11 scan-based wireless roaming process.

Related Work in Wireless Handoffs

Previous efforts for improving wireless handoffs can be categorized as either improved wireless protocols, handoff predictions, and vehicular handoffs. Our approach presented in the next section differs because we do not require modifying the wireless protocol and use much more fine-grain data-driven efforts for enabling the device to make more intelligent handoff decisions. Many of these efforts like faster handoffs are orthogonal to our efforts that could further optimize wireless performance.

Improved Handoff Protocols Significant modifications of the 802.11 AP handoff protocol can reduce the overheads of scans and promote more seamless switches between APs. In 802.11r Bangolae et al. [2006], the WiFi client's current AP provides the WiFi client with a set of APs to transition to. The WiFi client then directly negotiates the transition with its current AP before switching to the next AP for a seamless transition. Similar efforts modify the handoff protocol so that the WiFi client's current AP informs it about the next set of APs to consider when the device moves out of range Pack et al. [2007]. The WiFi client only performs a scan when it has exhausted the list of potential APs to consider. Another approach assigns time slots staggered by channel for APs to broadcast beacon frames Ramani and Savage [2005]. During the appropriate time slots, WiFi clients periodically listen on the corresponding channel to find observable APs. This enables continuous monitoring of surrounding APs, which effectively amortizes the cost of scans for more seamless switches between APs. Unfortunately, improved wireless protocols require significant modifications to all wireless devices, which makes these efforts impractical in the short term.

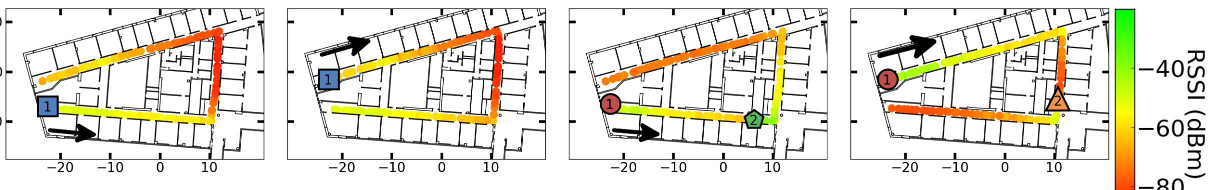
Handoff Prediction Alternate efforts better enables the WiFi client to make better wireless handoff decisions. Some efforts apply machine learning techniques to predict and trigger handoffs by looking at fine-grain metrics like RSSI changes and frequency of handoffs Javed et al. [2011]. Others anticipate handoffs and then react by multicasting data to nearby base stations in advance Seshan et al. [1997]. One can also react to handoffs by prefetching data before the WiFi client leaves a high bandwidth connection Drew and Liang [2004], Javed et al. [2011]. Most of these efforts use coarse-grain features that do not use fine-grain awareness of the surrounding environment. Efforts based on using location-based decision making rely upon fairly imprecise sensors like GPS outdoors and Place Lab indoors achieve a localization accuracy of only 20 to 30 meters Nicholson and Noble [2008].

Vehicular handoffs Unlike most traditional WiFi devices, vehicles move extremely fast, which makes AP handoffs even more difficult due to rapidly changing WiFi conditions Mahajan et al. [2007]. Some efforts focus on opportunistically taking advantage of fleeting WiFi connectivity. They employ techniques that focus on fast connections by reducing scan times and limiting retry timeouts Eriksson et al. [2008]. Other efforts focus on sustained WiFi connectivity for vehicles that remain within range of a private WiFi network. Instead of only using a single AP for connectivity, these efforts take advantage of nearby APs that help opportunistically forward wireless data to mitigate the effects of changing WiFi conditions Balasubramanian et al. [2008].

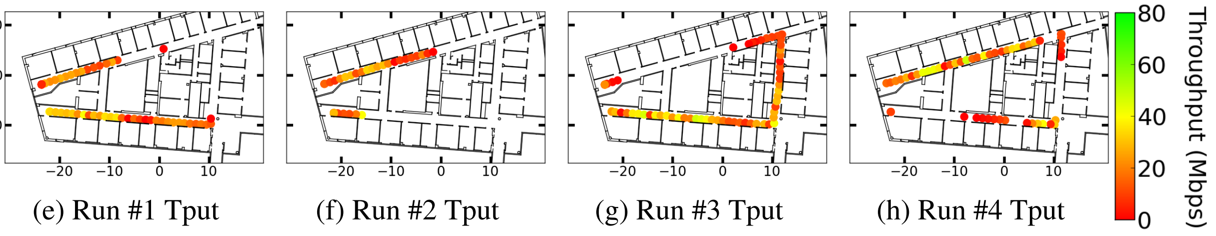
7.2.2 Reproducing Wireless Issues Due to Motion

In our initial investigation, our robot's connectivity issues seemed to arise sporadically. There were no specific locations where the robot always had no connectivity nor did the robot lose connectivity at the same location when moving across the same path. Our robot uses an off-the-shelf wireless device so these wireless challenges are not unique to robots. Unlike many mobile devices equipped with WiFi, our robot's unique abilities will allow us to track exactly how wireless performance varies over both time and space. We methodically subject the wireless device to identical motion in order to extract the similar patterns that lead to poor connectivity situations. Let us now show that tracking fine-grain variations enables us to confidently attribute our robot's actual wireless challenges to poor wireless handoff decisions.

We used a simple evaluation path that instructed the robot to follow a three-quarter loop path around three hallways in the environment, as shown in Figure 7.3. The path was chosen based on several factors. First, it needed to be long enough to ensure that the robot would definitely face connectivity issues. Next, we also instruct the robot to move both clockwise (Figure 7.3b and 7.3d) and counter clockwise (Figure 7.3a and 7.3c) directions to see if direction of motion plays a role. Finally, we ensure the robot traverses the environment at consistent speeds and covers each location at most once so that we can easily match wireless performance variations over time with 2D maps over space.

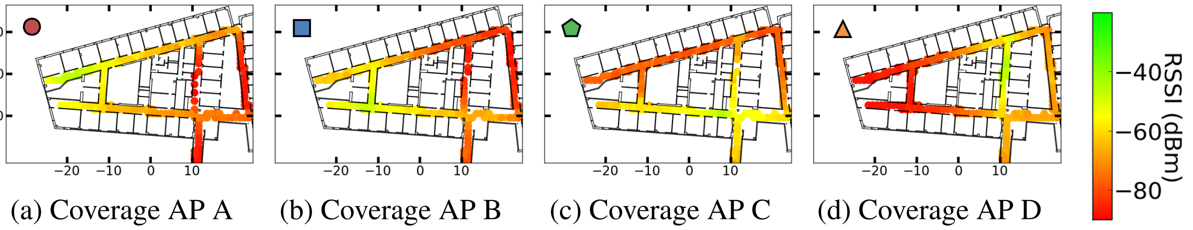


(a) Run #1 RSSI (b) Run #2 RSSI (c) Run #3 RSSI (d) Run #4 RSSI



(e) Run #1 Tput (f) Run #2 Tput (g) Run #3 Tput (h) Run #4 Tput

Figure 7.3: Poor wireless performance after moving across the same path four times.



(a) Coverage AP A (b) Coverage AP B (c) Coverage AP C (d) Coverage AP D

Figure 7.4: AP coverage to show range of available APs.

Figure 7.3 shows several samples of actual data collected by the robot as it was driven along the same path. During execution, the wireless device simultaneously captured RSSI, throughput, and the associated AP while the robot recorded timestamps and locations. Four noteworthy runs are shown in Figure 7.3 that shows RSSI (top), throughput (middle). In the RSSI maps (top), the unique shapes (squares, circles, and pentagons) reflect the location where the device first associated with the corresponding AP as identified by the color and shape. The numbered labels identify the order in which they were visited. We show AP coverage for each of these uniquely identified APs (bottom). Corresponding throughput while moving (middle) is also shown where large stretches of white space reflect absence of connectivity. Runs 1 (7.3a) and 3 (7.3a) began in the bottom left corner with the robot moving counterclockwise while Runs 2 (7.3b) and 4 (7.3d) started in the top left and moved clockwise. Numbered labels reflect the first point of association with each AP while the shape and color reflect a unique AP whose corresponding coverage map is shown in Figure 7.4.

7.2.3 Diagnosis as Poor Wireless Handoffs

Based on this data, we can conclude poor wireless performance stems from poor wireless handoffs. These failures stem from several poor wireless decisions: remaining associated with an AP that is already out of range, being too slow in switching away from an out-of-range AP, and poorly selecting which AP to switch to. Let us explain specifically how we make these conclusions based on these four runs.

Eliminating Possible Causes

With fine-grain tracking of wireless variations, we can rule out AP coverage since there are no clear dead spots. Also notice that RSSI varies gradually across several meters and eliminates the possibility of small and sudden dead zones. The direction of motion does not seem to be a factor because RSSI captured matches well with the corresponding AP coverage map, suggesting poor wireless performance must be occurring elsewhere. We cannot definitively rule out the effects of interference from other devices but wireless handoffs appear to be a more likely cause of our issues.

Poor AP Handoff Timing

There seems to be evidence supporting the hypothesis that associating with APs that are out of range is a big reason for the poor throughput. For example, run #1 and #2 remained associated with the same AP for the duration of the traversal and the wireless device naturally lacks connectivity when out of range. This suggests the need for switching APs as supported by run #3 and #4 that have connectivity in parts of the path where run #1 and #2 could not. However, the failure of all runs to sustain connectivity suggests that switches must be performed frequently and accurately.

Poor AP Selection

While both run #3 and #4 switch APs, the AP switch benefits run #3 more than #4 because run #4 switches to an AP that is already almost out of range. This likely stems from AP selection from stale measurements. Since scans for surrounding APs take several seconds, the device simply does not realize that the surrounding wireless environment has changed. We can confirm that this is not a coverage issue since there is at least one high RSSI AP along every location on the path.

The reason these connectivity issues are so prevalent for our robot is that few other devices stretch the boundaries of mobility (e.g. streaming real-time video while moving quickly); instead, most devices use wireless connectivity for portability (e.g. stationary data consumption at a desk). As a result, the robot is pushing the boundaries of wireless connectivity and revealing the limitations of wireless connectivity when subjected to continuous motion. We can see this manifest as poor timing of AP handoffs and poor decision in selecting APs to switch to. With fine-grain tracking of wireless performance over both space and time, we were able to clearly show that the robot was suffering from poor wireless handoffs.

7.3 Data-Driven WiFi Protocols for AP Handoffs

Our robot is clearly suffering from intermittent wireless connectivity due to poor wireless handoffs. Existing wireless handoff algorithms suffer from delayed and inaccurate decisions because they rely on reactive strategies based on immediate wireless measurements and scans. We propose an intelligent, data-driven handoff algorithm that uses fine-grain wireless maps to overcome our robot's poor wireless handoff problems. The intuition is that if the handoff algorithm uses knowledge about surrounding wireless conditions, then it will be able to better select APs and determine where they should occur. By improving the wireless handoff algorithm for the robot's wireless device, our approach works with the existing 802.11 WiFi protocol and does not require modifying other wireless devices.

As shown in Figure 7.5, our data-driven wireless handoffs use several pieces of information: where the robot is (localization), where it will go (navigation), and what are the surrounding wireless conditions (wireless snapshot). Access to an accurate wireless snapshot of current wireless conditions is essential (previously discussed in Section 4). In this section, we first introduce several handoff algorithms that differ in how they use fine-grain wireless maps. One notable handoff algorithm we focus on is the look-ahead planner that pre-plans an AP handoff plan. Finally, our evaluation shows that map-based handoffs significantly improve wireless performance over existing scan-based handoffs.

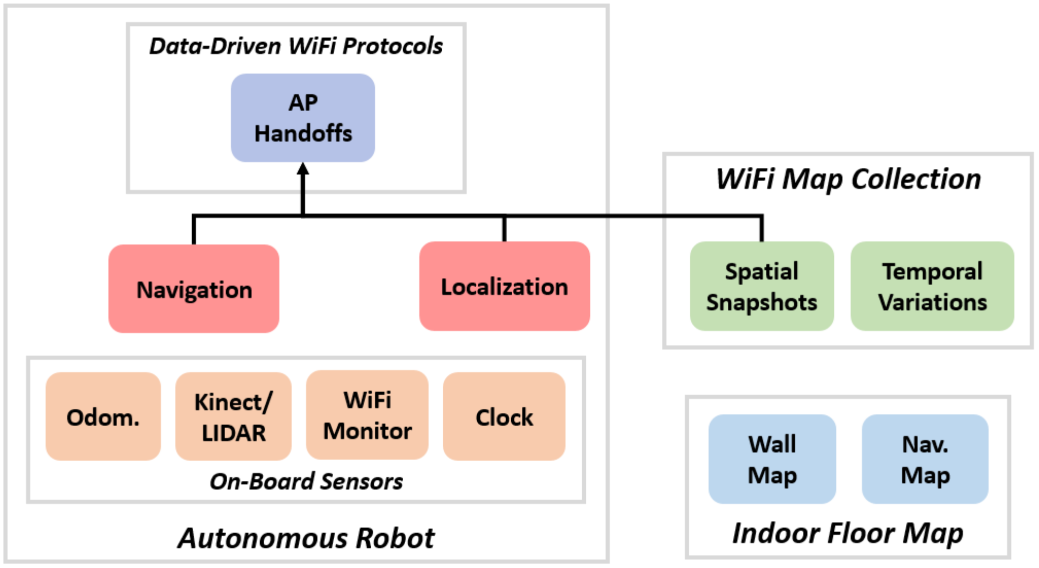


Figure 7.5: Data flow of our data-driven wireless handoffs algorithm.

7.3.1 Wireless Map Based Handoff Algorithms

Wireless handoff algorithms make two key decisions: when to disassociate with an AP and which AP to select next. We present several variations of informed wireless handoff algorithms to see what information and switching strategies result in the most significant improvements in wireless performance. The four algorithms in order of increasing complexity are: **aggressive disassociation**, **location-based AP selection**, **highest RSSI AP policy**, and **look-ahead planning** as summarized in Table 7.1. We first highlight key differences between these handoff algorithms and then discuss each in more detail.

Table 7.1: Table of handoff algorithms.

Handoff Algorithm	**AP Dissasociation**	**AP Selection**
Aggressive Disassociations	When RSSI falls below threshold	Issue wireless scan for highest RSSI AP
Location-Based AP Selection	When RSSI falls below threshold	Query wireless map for highest RSSI AP at current location
Highest RSSI AP Policy	When current AP is not the highest RSSI AP from wireless map	Query wireless map for highest RSSI AP at current location
Look-Ahead Plan	When arriving at next waypoint location in AP handoff plan	Query AP handoff plan for current waypoint AP

Aggressive disassociations serves as a baseline to compare against traditional reactive, scan-based algorithms discussed earlier in Section 7.2.1. The key difference is that we lower the threshold RSSI for disassociating to test the hypothesis that perhaps the threshold RSSI is not tailored for extremely mobile devices. Our next three algorithms are more informed because they combine the robot's location with the wireless map. Location-based AP selection evaluates the impact of using a wireless map for selecting APs but continues to use a reactive threshold for disassociating. Higher RSSI AP policy relies completely on location and the wireless map for both disassociation and selection decisions. Finally, look-ahead planning incorporates the device's pre-planned navigation route to further optimize where and when wireless handoffs occur.

Aggressive Disassociation This algorithm shown in Algorithm 6 mimics the default, scan-based handoff algorithm (Section 7.2.1) that disassociates when RSSI of the current AP falls below some threshold. Given strong AP coverage across our environment, we use an aggressive RSSI threshold T_{RSSI} of -70 dBm. This threshold is chosen based on the RSSI histogram in Figure 4.8 that showed consistently high RSSI across most locations in our environment. This serves as a baseline and disabuses the idea that more aggressive disassociations would resolve the wireless connectivity issues.

Algorithm 6 *Aggressive Disassociations:* disassociate when RSSI below threshold and then scan for APs

```
Require: Current state $S$ with RSSI of associated AP $\mathbf{S_{RSSI}}$. Given a minimum threshold RSSI
    with AP, $\mathbf{T_{RSSI}}$.
1: function AGGRESSIVE($S, T$)
2:     if $S_{RSSI} \leq T_{RSSI}$ then
3:         $APs \leftarrow APScan()$
4:         $AP_{curr} \leftarrow max_{RSSI}(APs)$
5:         $Connect(AP_{curr})$
6:     end if
7: end function
```

Location-Based AP Selection Instead of scanning for surrounding APs, this algorithm shown in Algorithm 7 queries the given wireless map for the best available AP at the device's current location. By using a wireless map, this approach removes the uncertainty of scans by providing the device with the actual highest RSSI AP at its current location. We continue to use the threshold-based disassociations to better isolate the impact of more intelligent AP selection and reduced overheads of scans on wireless performance. Our evaluation will show that the most significant wireless performance improvement arises from using the wireless map to select which AP to associate with.

Highest RSSI AP Policy A device that is continuously aware of its location and has access to accurate wireless maps does not need to react to immediately collected wireless measurements so we present full, location-based handoffs. This algorithm shown in Algorithm 8 continuously queries the highest RSSI AP based on its current location and switches whenever they do not

Algorithm 7 *Location-Based AP Selection:* aggressively disassociate and then select AP using a wireless map

```
Require: Current state $S$ with RSSI of associated AP $\mathbf{S_{RSSI}}$ and device's location $\mathbf{S_{LOC}}$. Given
    a minimum threshold RSSI with AP, $\mathbf{T_{RSSI}}$. Previously collected wireless map $\mathbf{M_{WiFi}}$
1: function INFORMED($S, T, M$)
2:     if $S_{RSSI} \leq T_{RSSI}$ then
3:         $APs \leftarrow APsAtLocation(S_{LOC}, M_{WiFi})$
4:         $AP_{curr} \leftarrow max_{RSSI}(APs)$
5:         $Connect(AP_{curr})$
6:     end if
7: end function
```

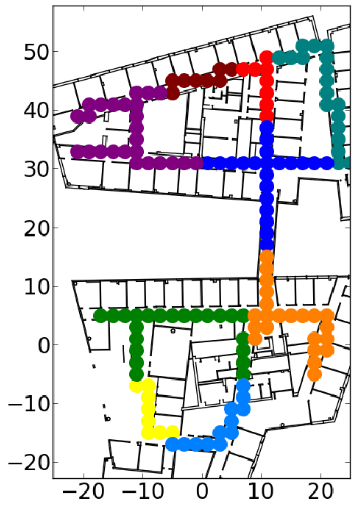


Figure 7.6: Highest RSSI AP per Region.

match. Notice there is no need for a threshold RSSI for disassociation as this decision is now driven directly from the wireless map. An example wireless map showing the highest RSSI AP across our environment is shown in Figure 7.6. This simple algorithm avoids the need to specify disassociation thresholds and AP selection is simply. This approach focuses on the importance of having the highest RSSI at all times. Our evaluation will show this algorithm introduces excessively frequent handoffs that are significant due to the non-negligible overheads of performing handoffs.

Algorithm 8 *Highest RSSI AP Policy:* continuously associate with the highest RSSI AP using a wireless map

```
Require: Current state $S$ with device's location $\mathbf{S_{LOC}}$ and current AP $\mathbf{S_{AP}}$. Previously collected
    wireless map $\mathbf{M_{WiFi}}$.
1: function RADIO($S, M$)
2:     $APs \leftarrow APsAtLocation(S_{LOC}, M_{WiFi})$
3:     $AP_{curr} \leftarrow max_{RSSI}(APs)$
4:     if $S_{AP} \neq AP_{curr}$ then
5:         $Connect(AP_{curr})$
6:     end if
7: end function
```

7.3.2 Look-Ahead Planning

The look-ahead planner is a unique algorithm tailored for autonomous robots that plan their movements in advance. Unlike the previous algorithms, the look-ahead planner has two steps: an offline handoff planner that pre-computes which AP to associate with along its path and an online execution of this handoff plan when the device moves. The offline planner is shown in Algorithm 10 where the goal is to minimize the total number of handoffs along the device's navigation path while ensuring strong, continuous connectivity with nearby APs. An example of the pre-computed AP assignment plan used in our evaluation is shown in Figure 7.7b. The online execution of the plan is shown in Algorithm 9, where the algorithm tracks and reacts to changing AP assignments as it passes waypoint locations along its path. Although this two-step algorithm is more complex than the others, our evaluation will show that it achieves the best wireless performance.

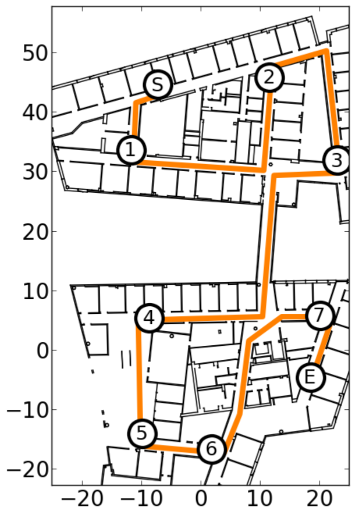
Pre-computing the AP Handoff Plan

The handoff planner has two steps: computing the minimum AP handoff plan and then further refining the plan to optimize where the handoffs are executed along the device's path. The algorithm for pre-computing the minimum AP handoff plan is shown in Algorithm 10. The function $genHandoffPlan$ is given the robot's future path, wireless map, and a minimum threshold

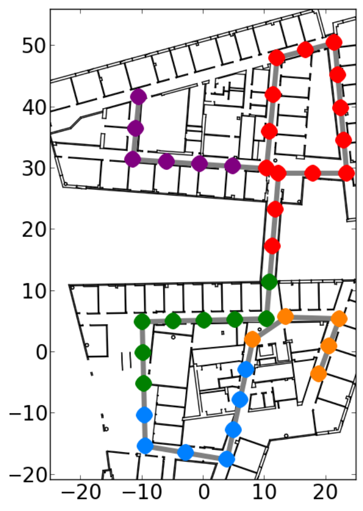
Algorithm 9 *Look-Ahead Plan:* follow a precomputed a sequence of AP transitions along a known path.

```
Require: Current state $S$ with device's location $\mathbf{S_{LOC}}$. Given a precomputed queue $\mathbf{Q_{LOC,AP}}$
    of AP assignments for each location along device's given planned path from Algorithm 10.
1: function LOOKAHEAD($S, Q$)
2:     if $isAtLocation(Q_{LOC}, S_{LOC})$ then
3:         $AP_{curr} \leftarrow Q_{AP}$
4:         $Connect(AP_{curr})$
5:         $Q \leftarrow Q.next()$
6:     end if
7: end function
```

RSSI. The path is represented as a sequence of discrete waypoint locations (2m between consecutive waypoints in our case). Using the wireless map, we first compute the set of APs that exceed a minimum threshold RSSI for each location along the specified path. Next, we perform a breadth first search expansion of an AP graph to find the minimum number of AP handoffs that ensure connectivity along the entire path. We reach the goal when we have a plan that assigns an AP for all waypoint locations along the device's navigation path.



(a) Navigation path



(b) Look-Ahead Handoff Plan

Figure 7.7: Look-ahead planner computes an AP handoff plan for a given path.

Generating Minimum Handoff Plan

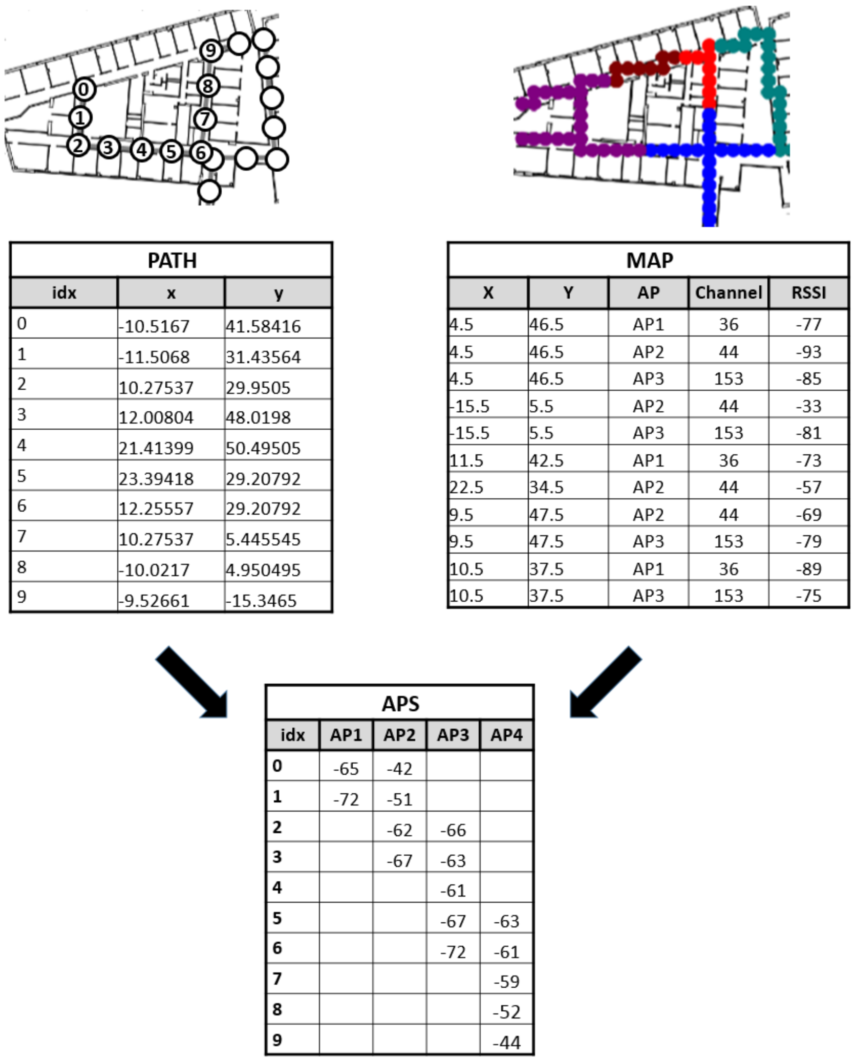
Figure 7.8 presents a visualization of the navigation PATH represented as a sequence of waypoint locations and the wireless MAP represented as a set of access point parameters (id, channel, rssi)

Algorithm 10 Pre-computing an AP handoff plan based on the device's future path.

Require: Device's PATH, wireless MAP, minimum threshold RSSI.

```
1: function GENHANDOFFPLAN(PATH, MAP, RSSI)
2:     idx$_{start}$ ← $0$
3:     idx$_{end}$ ← PATH.$length$
4:     APS ← {}
5:     for $i$ in $range$(idx$_{start}$, idx$_{end}$) do
6:         APS[$i$] ← $getAPsAtLoc$(PATH[$i$], MAP, RSSI)
7:     end for
8:
9:     NODES$_{curr}$ ← []
10:    for $AP$ in APS[idx$_{start}$] do
11:        node$_{init}$ ← $newNode$(idx$_{start}$, $AP$)
12:        NODES$_{curr}$.$add$(node$_{init}$)
13:    end for
14:
15:    while $True$ do
16:        NODES$_{next}$ ← []
17:        for node$_{curr}$$in$NODES$_{curr}$ do
18:            ap$_{curr}$ ← $getAP$(node$_{curr}$)
19:            idx$_{curr}$ ← $getIdx$(node$_{curr}$)
20:            idx$_{next}$ ← $getNextIdxfromAP$(ap$_{curr}$,idx$_{curr}$, APS)
21:            if idx$_{next}$ == idx$_{end}$ then
22:                return node$_{curr}$
23:            end if
24:            for $AP$ in APS[idx$_{next}$] do
25:                node$_{next}$ ← NODE$_{curr}$.$step$(idx$_{next}$, $AP$)
26:                NODES$_{next}$.$add$(node$_{next}$)
27:            end for
28:        end for
29:        NODES$_{curr}$ ← NODES$_{next}$
30:    end while
31: end function
```

for each grid region introduced earlier in Chapter 4. From a simple matching based on location, we compute APS that determines which access points are available along each waypoint of the path. This corresponds to lines 5-7 of Algorithm 10. We filter out access points that fall below a target threshold RSSI because the entries of APS should represent access points that would provide strong connectivity.



PATH		
idx	x	y
0	-10.5167	41.58416
1	-11.5068	31.43564
2	10.27537	29.9505
3	12.00804	48.0198
4	21.41399	50.49505
5	23.39418	29.20792
6	12.25557	29.20792
7	10.27537	5.445545
8	-10.0217	4.950495
9	-9.52661	-15.3465

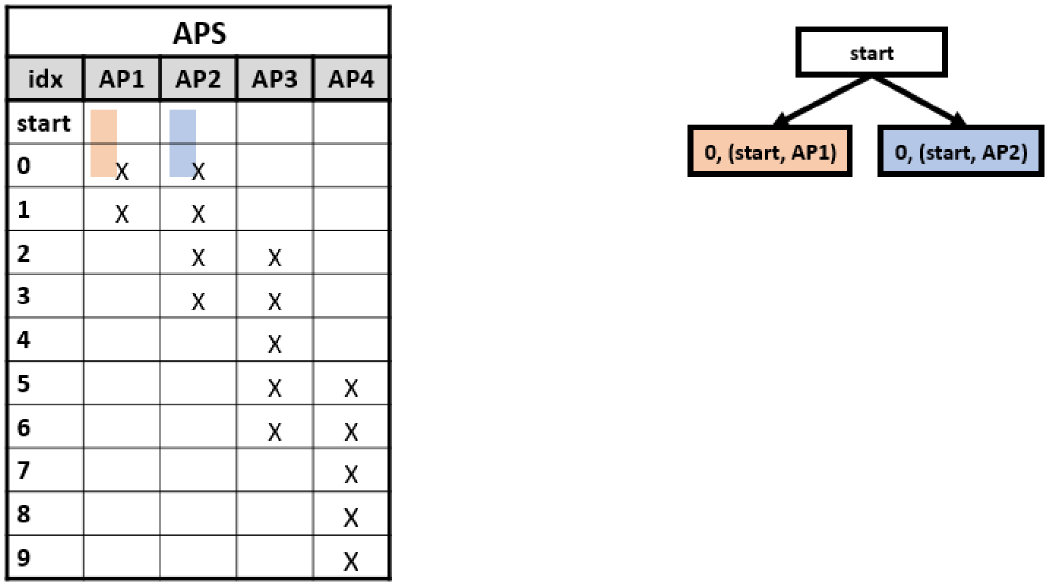
MAP				
X	Y	AP	Channel	RSSI
4.5	46.5	AP1	36	-77
4.5	46.5	AP2	44	-93
4.5	46.5	AP3	153	-85
-15.5	5.5	AP2	44	-33
-15.5	5.5	AP3	153	-81
11.5	42.5	AP1	36	-73
22.5	34.5	AP2	44	-57
9.5	47.5	AP2	44	-69
9.5	47.5	AP3	153	-79
10.5	37.5	AP1	36	-89
10.5	37.5	AP3	153	-75

APS				
idx	AP1	AP2	AP3	AP4
0	-65	-42		
1	-72	-51		
2		-62	-66	
3		-67	-63	
4			-61	
5			-67	-63
6			-72	-61
7				-59
8				-52
9				-44

Figure 7.8: Determining available APs along path waypoints by using a WiFi map.

The plan is generated using a forward tree search across sequences of possible AP handoffs. Each node $(\text{idx}, \text{AP}[i])$ in this tree is assigned a waypoint index along the device's path as well as the corresponding access point it should be transitioning to at this step. Our visualizations more explicit each node as $(\text{idx}, \text{AP}_{from}, \text{AP}_{to})$

Initially, we populate the search tree with APs that can be reached from the starting waypoint location as shown in line 9-13 of Algorithm 10. This is visualized in Figure 7.9a where the initial AP tree is populated with two nodes. On the left, APS illustrates the set of access points available at each waypoint location. Therefore, the initial two nodes reflect the possibility of either associating with AP1 or AP2 at waypoint 0. The colored lines in APS correspond to each leaf node in the tree. It will help to highlight how far the search has proceeded along each unique branch.



APS				
idx	AP1	AP2	AP3	AP4
start				
0	X	X		
1	X	X		
2		X	X	
3		X	X	
4			X	
5			X	X
6			X	X
7				X
8				X
9				X

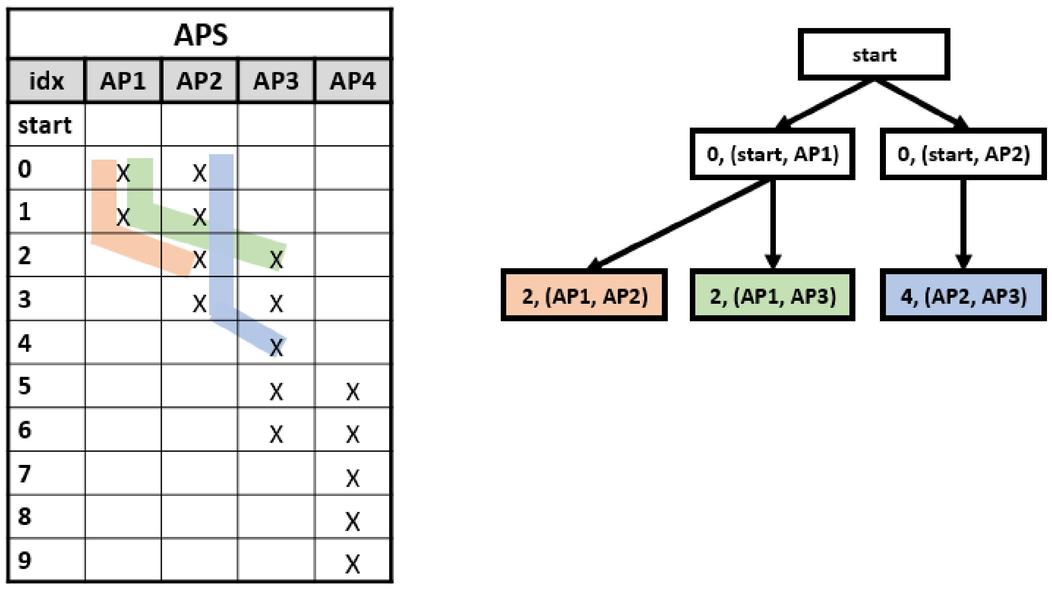
(a) Initial search tree: identifies APs available at the first path waypoint.

Subsequent steps of the search correspond to line 16-29 of Algorithm 10. This is a simple breadth first search expansion of the tree based on possible AP handoffs. In Figure 7.10a, we explore the two possible branches. If we start off from waypoint 0 associated with AP1, we can remain associated until waypoint 2 where we have the option to then switch to either AP2 or AP3. We can see how this naturally results in two additional nodes. If we start off from waypoint 0 associated with AP2, we are not forced to switch until waypoint 4, where there is only one option. Note that each level of our AP tree reflects the number of handoffs performed. We are not performing an expansion of unique AP assignments across each waypoint, which constrains the size of the search space.

We proceed through subsequent steps of the expansion until we reach a leaf node(s) that achieves an AP assignment for all waypoints. In Figure 7.11, we find two possible plans that require the same number of handoffs. Both of these plans are preferred to the alternate branch that will require more than three handoffs. With this forward expansion, we have quickly narrowed the set of possible AP handoffs the device can perform while following the given path.

Optimizing Handoff Points

Two challenges remain from our minimum handoff planner: timing where the handoff should occur and choosing among several possible handoff plans. While all plans generated will ensure conenctivity above the given threshold RSSI, we have the opportunity to further refine the plan



APS				
idx	AP1	AP2	AP3	AP4
start				
0	X	X		
1	X	X		
2		X	X	
3		X	X	
4			X	
5			X	X
6			X	X
7				X
8				X
9				X

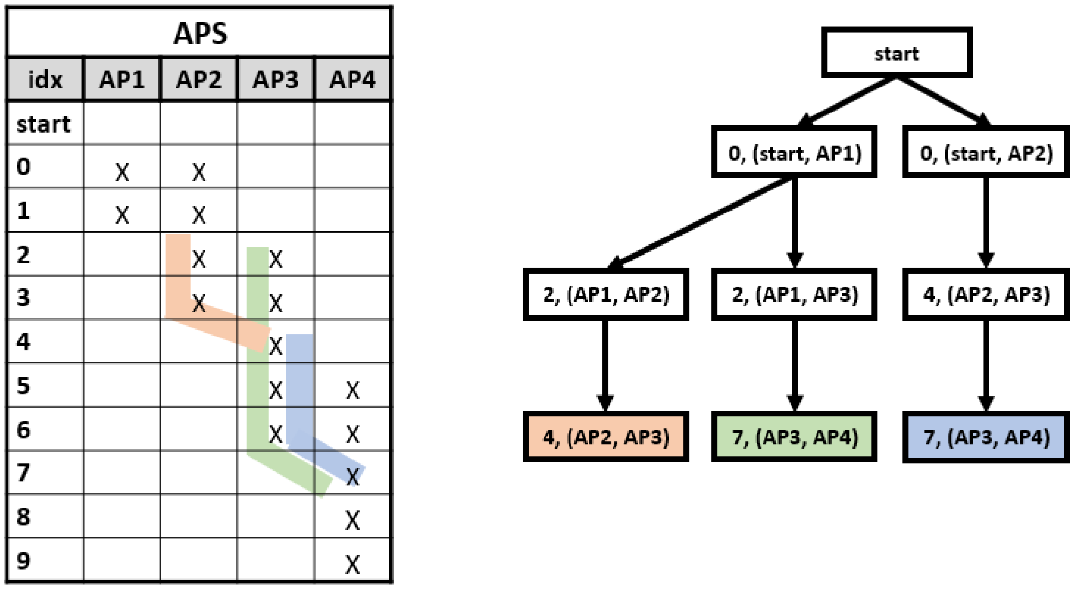
(a) After one step: based on possible initial AP assignments, determine how far device can remain connected to AP

Figure 7.10: Initial breadth first expansion of AP graph tree

to increase RSSI along the overall path. For this step, we take the sequence of AP switches from each plan and try to optimize the waypoint where the handoff will occur based on RSSI. In addition, we can use RSSI across the entire path to identify the best available handoff plan when we have multiple options.

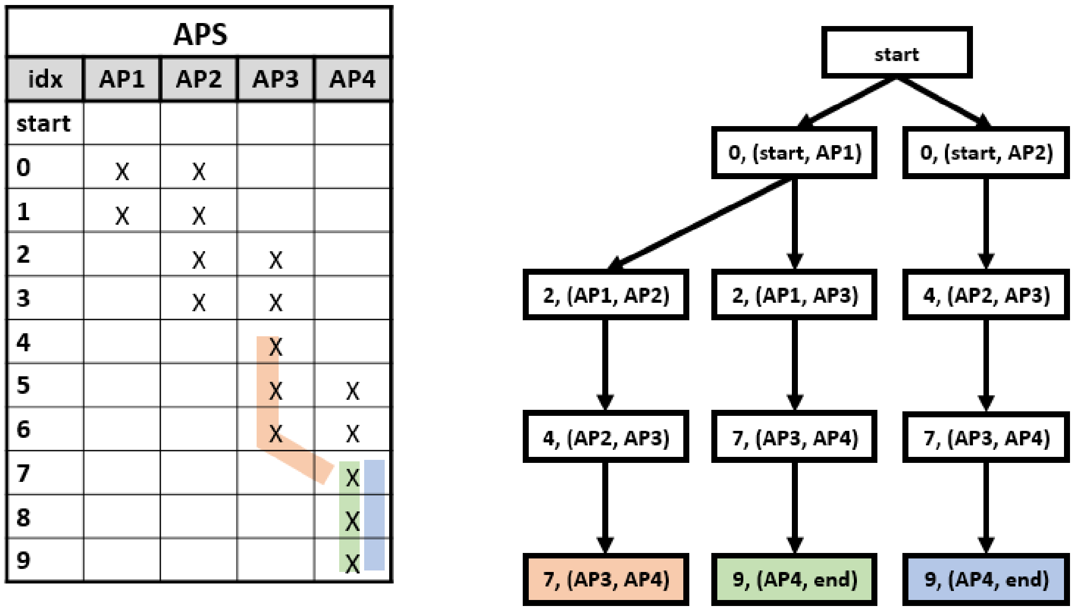
Figure 7.12 shows how we further refine the two possible minimum handoff plans found above. We first take the original plan and expand it into AP assignments along the entire set of navigation waypoints. From this plan, we can easily determine the source AP and destination AP for each waypoint. A simple lookup will tell us what the corresponding RSSI is for each of these two AP choices. The optimal AP assignment is allows to transition early to the destination AP when RSSI is higher than the source AP. Notice that for both plans, the final optimized plan has higher overall RSSI than the original.

The same idea allows us to choose the best plan when several choices exist. There are many different metrics one could use to differentiate these paths. In this example, we can see that AP plan #2 sustains much higher overall RSSI along the entire path making it a better choice. Future work could use this same approach to enable the device to select between several possible navigation paths based on wireless connectivity requirements. For example, our robot's navigation paths are generated based on shortest distance. If the robot wanted to optimize for connectivity, the robot might prefer slightly longer paths that result in much fewer handoffs.



APS				
idx	AP1	AP2	AP3	AP4
start				
0	X	X		
1	X	X		
2		X	X	
3		X	X	
4			X	
5			X	X
6			X	X
7				X
8				X
9				X

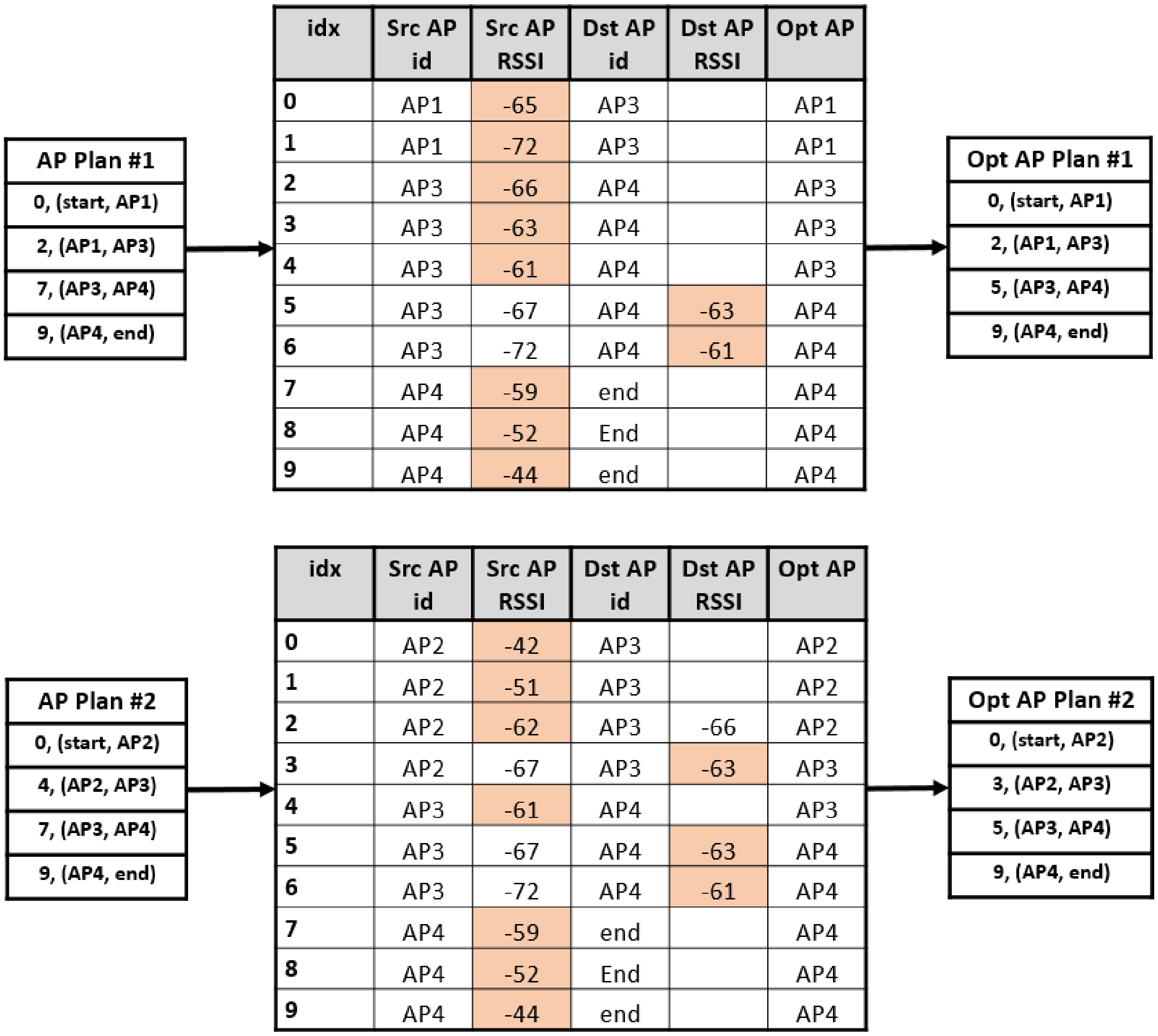
(a) After two steps: continues forward search for AP handoff options



APS				
idx	AP1	AP2	AP3	AP4
start				
0	X	X		
1	X	X		
2		X	X	
3		X	X	
4			X	
5			X	X
6			X	X
7				X
8				X
9				X

(b) After three steps: two possible AP handoff plans for the device’s entire path

Figure 7.11: Subsequent steps showing breadth first expansion of AP handoff tree



AP Plan #1
0, (start, AP1)
2, (AP1, AP3)
7, (AP3, AP4)
9, (AP4, end)

idx	Src AP id	Src AP RSSI	Dst AP id	Dst AP RSSI	Opt AP
0	AP1	-65	AP3		AP1
1	AP1	-72	AP3		AP1
2	AP3	-66	AP4		AP3
3	AP3	-63	AP4		AP3
4	AP3	-61	AP4		AP3
5	AP3	-67	AP4	-63	AP4
6	AP3	-72	AP4	-61	AP4
7	AP4	-59	end		AP4
8	AP4	-52	End		AP4
9	AP4	-44	end		AP4

Opt AP Plan #1
0, (start, AP1)
2, (AP1, AP3)
5, (AP3, AP4)
9, (AP4, end)

AP Plan #2
0, (start, AP2)
4, (AP2, AP3)
7, (AP3, AP4)
9, (AP4, end)

idx	Src AP id	Src AP RSSI	Dst AP id	Dst AP RSSI	Opt AP
0	AP2	-42	AP3		AP2
1	AP2	-51	AP3		AP2
2	AP2	-62	AP3	-66	AP2
3	AP2	-67	AP3	-63	AP3
4	AP3	-61	AP4		AP3
5	AP3	-67	AP4	-63	AP4
6	AP3	-72	AP4	-61	AP4
7	AP4	-59	end		AP4
8	AP4	-52	End		AP4
9	AP4	-44	end		AP4

Opt AP Plan #2
0, (start, AP2)
3, (AP2, AP3)
5, (AP3, AP4)
9, (AP4, end)

Figure 7.12: Differentiating possible handoff plans and further refining where handoffs occur.

7.4 Evaluating Impact of Handoffs on WiFi Performance

Our data-driven AP handoffs significantly improve wireless performance while moving. We show how wireless performance is affected by different variations in our proposed handoff algorithms. We first magnify fine-grain variations in wireless performance across a single, identical path to reveal differences across the handoff algorithms. We then perform a thorough comparison across many repeated executions of scan-based versus location-based handoffs at three different speeds. The result is significant improvements in wireless performance while moving because fine-grain sensor measurements enable accurate, real-time decision making.

7.4.1 Fine-Grain Improvements in Wireless Performance

To show wireless performance differences across the four different handoff algorithms, we subject the wireless device to identical traversals of a complex path that requires the device to make many challenging AP handoffs. The path is shown in Figure 7.7a with starting location (S) and end location (E) marked. The numbers indicate the order that intermediate waypoints are visited. The device traverses each location at most once to make it easier to inspect performance variations with wireless maps.

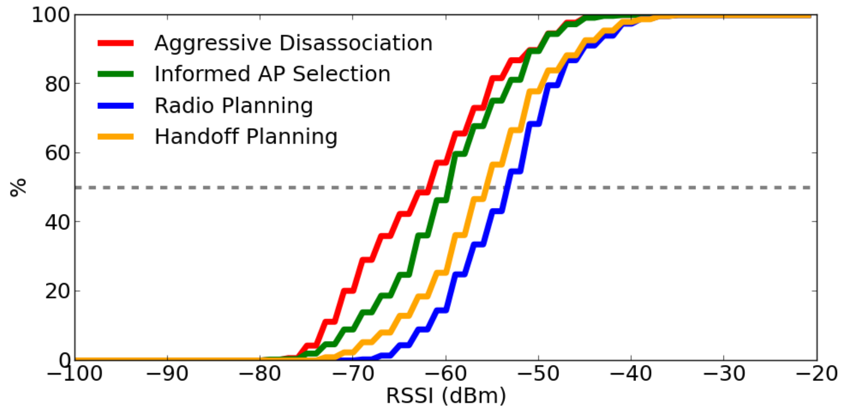
Overall performance for one iteration is summarized in Table 7.2. We see that **location-based AP selection** actually minimizes the number of AP handoffs, **highest RSSI policy** ensures highest median RSSI, and **look-ahead planning** achieves highest median throughput. Unsurprisingly, all three of our location-based handoffs clearly dominate the scan-based efforts. This suggests that we can significantly improve wireless performance by eliminating the overheads of wireless scans and intelligently selecting APs. As we will show, more complex algorithms using more information can continue to improve wireless performance.

	Aggressive Disassociations	Informed AP Selection	Radio Planning	**Handoff Planning**
Mean RSSI (dBm)	-61.8	-59.4	**-53.1**	-55.3
Med RSSI (dBm)	-61.0	-59.0	**-53.0**	-55
Mean Tput (Mbps)	29.13	41.47	43.60	**45.44**
Med Tput (Mbps)	33.00	46.00	46.53	**48.51**
# Gaps	6	**4**	8	5
Gap Time (s)	35.50	10.62	18.30	**10.26**

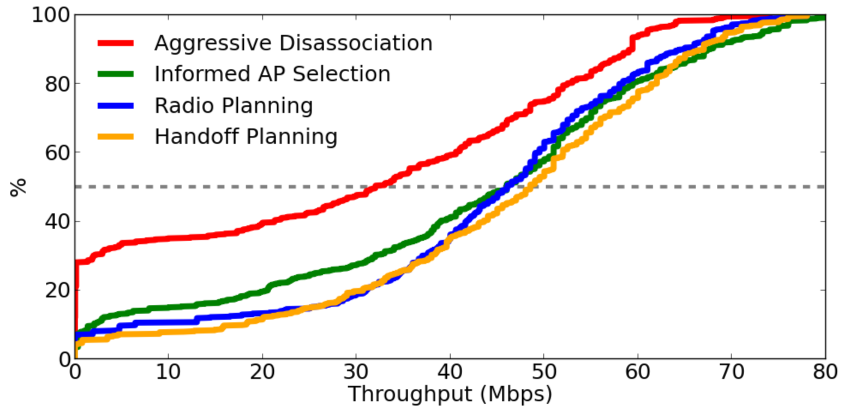
Table 7.2: Summary of aggregate measurements across four handoff algorithms.

Figure 7.14 and 7.15 shows variations in wireless performance by time and locations. On top, we show how RSSI varies spatially while moving to different locations. We show corresponding variations over time for both RSSI (middle) and throughput (bottom). The labeled numbers correspond to the location waypoints on top with both time-varying RSSI (middle) and throughput (bottom) over time. RSSI of -65 dBm is marked with a gray line as a reference.

There are several key takeaway from this data. First, we can see exactly why reactive, scan-based efforts fare poorly. Notice that **aggressive disassociations** experiences two lengthy periods without connectivity that we see corresponds to RSSI lingering at levels of low RSSI. This is a result of poor AP selection from noisy scans. Next, reactively deciding to disassociate based on RSSI thresholds cannot achieve the level of wireless performance that knowledge about the wireless surroundings can provide. This is shown by **highest RSSI AP policy** and **look-ahead planning** being most successful at ensuring consistently high RSSI across the entire path. Finally, there are clearly overheads in switching APs so reducing the number of handoffs is equally important. **Look-ahead planning** outperforms **highest RSSI AP policy** despite experiencing slightly lower RSSI due to much fewer handoffs.



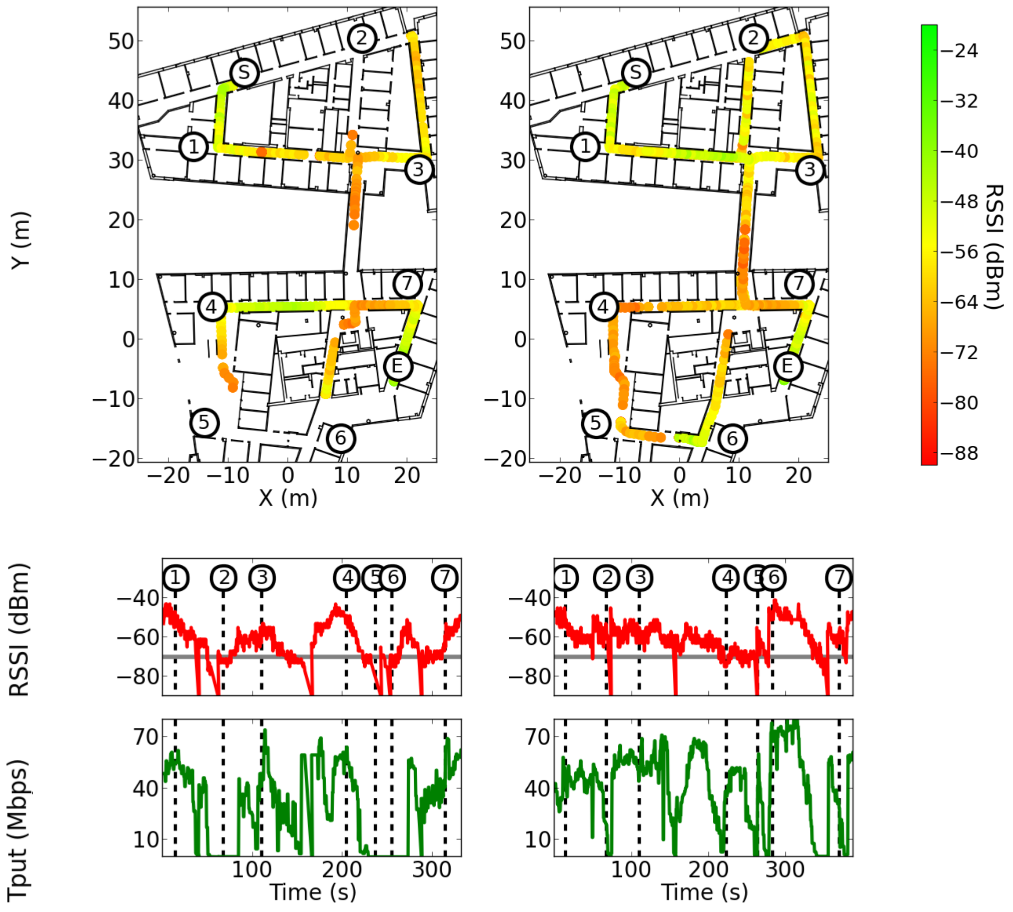
(a) CDF of RSSI



(b) CDF of throughput

Figure 7.13: Comparing aggregate WiFi performance along the same path.

These detailed measurements show why location-based handoffs are able to overcome systemic problems with scan-based efforts. We also see further opportunities for optimizing handoffs. For example, throughput in some cases is higher despite lower RSSI, suggesting other factors like congestion or interference may play a role. In addition, we can see for **location-based AP selection** how less frequent handoffs sometimes results in less jittery throughput so the duration of handoffs may be an important consideration in the future.



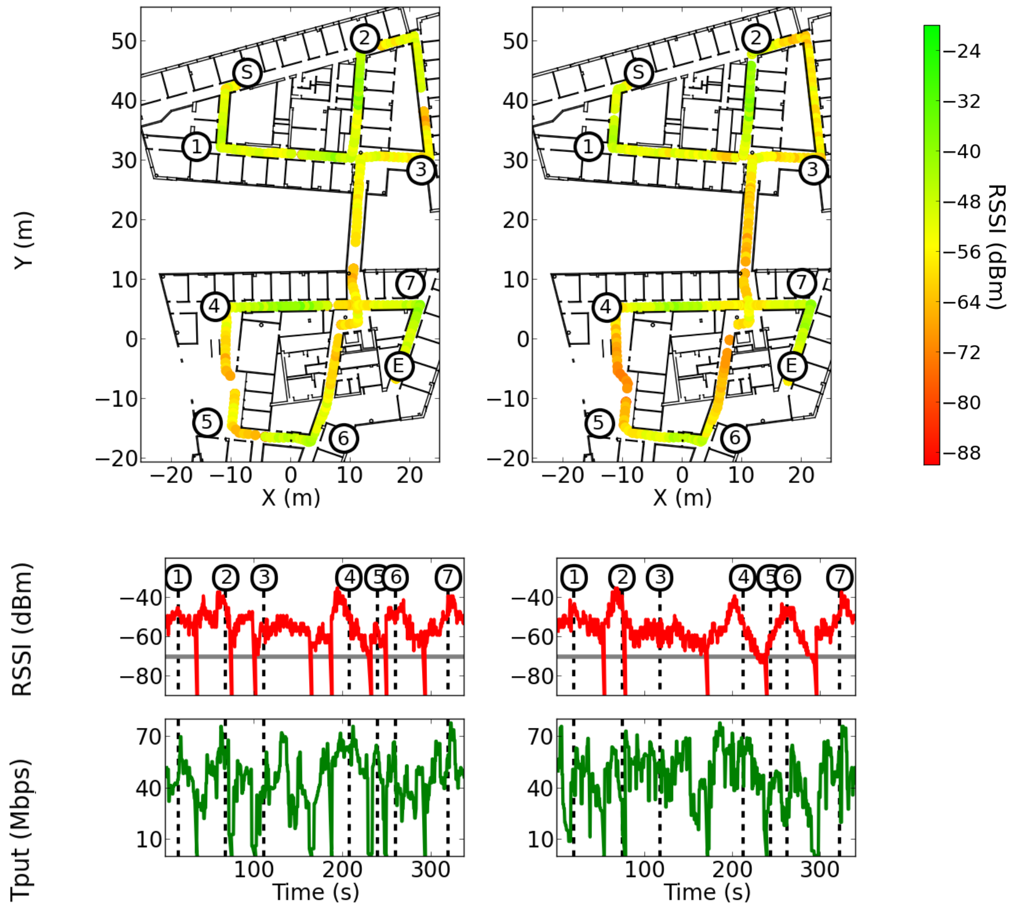
(a) Aggressive Disassociation (b) Location-Based AP Selection

Figure 7.14: Tracking wireless performance along same path.

7.4.2 Aggregate Improvements in Wireless Performance

We perform a robust comparison of scan-based and location-based handoff algorithms under continuous motion at several different speeds. In particular, we compare **aggressive disassocations** against **highest RSSI policy**. We select a 150 m path and perform four iterations of each speed and handoff algorithm combination for a total of 3.6 kilometers. The length of each path was chosen such that both algorithms could be run sequentially for every speed before the robot's battery needed to be charged to minimize the impact of the environment changing over time.

Figure 7.16 shows a cumulative distribution function (CDF) showing the distribution of how throughput varies along the path. The entire path was divided into 2 meter segments for which



(a) Highest RSSI Policy (b) Look-Ahead Planning

Figure 7.15: Tracking wireless performance along same path.

we computed the average throughput. The CDF shows the cumulative throughput for each of these segments. Notice the scan-based efforts do not have wireless connectivity for significant proportions of the path due to a combination of 3-5 second overheads for scans and increasingly frequent handoffs due to poor AP selection. Even when moving at the slowest speed, scan-based efforts are without connectivity for 10% of the path, which is intolerable for applications like telepresence. WiFi performance for scan-based handoffs improves as the device moves more slowly due to less frequent changes in surrounding WiFi conditions.

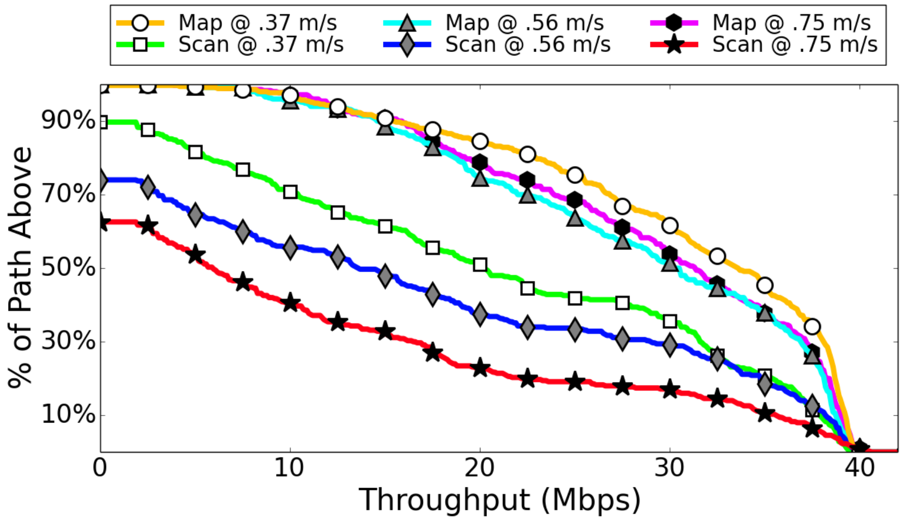


Figure 7.16: CDF showing the % of path above some throughput for different speeds.

Our data-driven wireless handoffs have superior cumulative WiFi performance at any speed. This shows that many of the interruptions in connectivity for scan-based handoffs arise from a combination of scan overheads and poor AP selection. As we can see, eliminating scans entirely and more intelligently selecting access points can significantly improve wireless performance. We showed that data-driven wireless handoffs can ensure reliable wireless connectivity for devices like mobile robots with strenuous wireless demands. Informed handoffs are also a pragmatic solution as they do not require modifying hardware or software on surrounding wireless devices and work seamlessly with the existing 802.11 WiFi protocol.

7.4.3 Discussion

Our evaluation results only shows marginal wireless performance improvements for the look-ahead planner when compared to the other handoff algorithms. There are scenarios where the look-ahead planner will outperform the others due to additional foresight about where the device will move in the future. This allows the look-ahead planner to generate handoff plans that more effectively break ties when selecting between possible access points and choosing access points that reduce the total number of handoffs.

Breaking Ties Between Access Points Suppose we are faced with the scenario shown in Figure 7.17 where the device is at point P1 and there are three equidistant access points. Except for the look-ahead planner, all other handoff algorithms would have no way to differentiate them due to identical RSSI measurements. Suppose that we know the device will be moving towards AP2. In this situation, the device should select AP2 so as to avoid an additional handoff. This is only possible with the look-ahead planner that can break ties by looking forward into the future based on what subsequent handoffs will be required in the future.

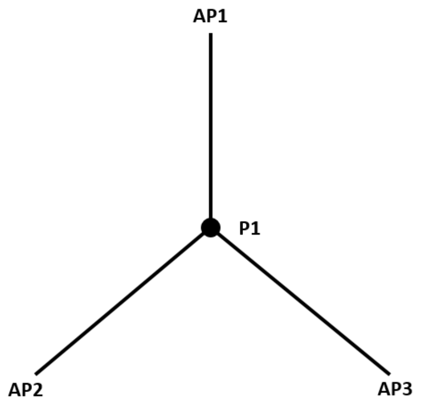
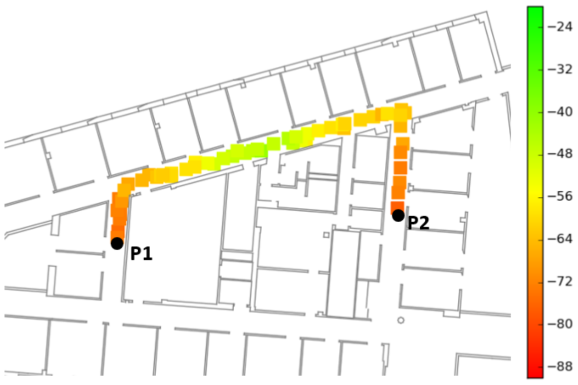


Figure 7.17: Scenario in which all access points appear equally good

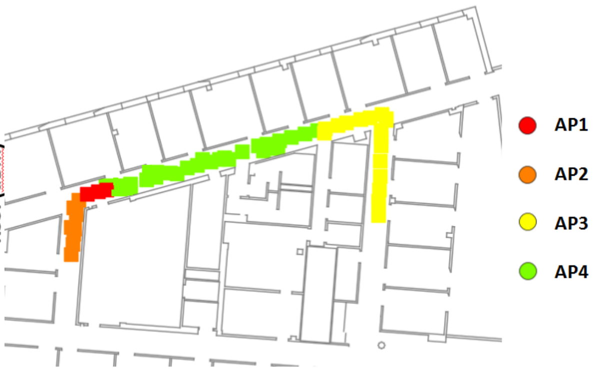
Selecting Access Points that Reduce Handoffs Suppose that the wireless device follows the path shown in Figure 7.18a by moving repeatedly from point P1 to P2. We can see that AP2 fully cover this path but that signal strengths are quite low around points P1 and P2. Only the look-ahead planner has the opportunity to realize that it can avoid any handoffs by remaining associated with AP2. The handoffs that disassociate when RSSI falls below some given threshold are likely experience momentarily low RSSI at the endpoints P1 and P2, resulting in a subsequent disassociation. Because these reactive efforts do not know that the device will immediately turn back around, they will end up performing excessive handoffs. Similarly, as shown in the wireless map in Figure 7.18b, the highest RSSI AP policy would switch excessively along this path with a total of four handoffs for each traversal. Only the look-ahead planner has the necessary foresight.

7.5 Summary

Fine-grain sensor measurements collected by mobile devices can be extremely valuable for overcoming a wide range of problems. Not only did we diagnose challenging motion-based wireless



(a) Coverage Map for AP2



(b) Highest RSSI AP along Path

Figure 7.18: Sample path with corresponding wireless conditions

connectivity issues by recreating the problematic situations but we also resolved them by introducing wireless map-based solutions so that the robot could make accurate and timely handoffs decisions. We were able to identify and propose a solution to significantly improve our own autonomous robot's intermittent wireless connectivity issues in a real enterprise wireless environment. In the future, there are opportunities to extend these data-driven efforts to further differentiate navigation paths based on alternate metrics like wireless connectivity.

Chapter 8

Conclusion and Future Work

In this chapter, we summarize the contributions of this thesis, describe opportunities for future work, and recap key takeaways from this work.

8.1 Contributions

The key contributions are:

Data Collection with Mobile Platforms We introduce a data collection framework that collects sensor measurements while moving. We specifically investigate how specific sensors behave when collected by a moving robot. We show the key limitation of data collection with mobile platforms is measurement sparsity when drilling down by specific factors that subsequent contributions try to address.

Spatio-Temporal Analysis of Fine-grain Sensor Measurements We present a discrete spatio-temporal representation to address the challenge of sensor measurements captured across different locations and times. Navigation adjusted grids spatially segmented the environment based on regions where the robot can actually reach to ensure better spatial distribution of sensor measurements. We show this discretization is able to reveal fine-grain insights about our surrounding environments.

Navigation Trajectory Optimized for Sensor Measurement Needs Mobile platforms have limited data collection opportunities making it important to develop intelligent navigation strategies. We explore two types of active navigation strategies: one where the robot navigates naturally along hallways and another where the robot moves to specific small grid regions in a more zig-zag manner. We develop algorithms that effectively utilize the robot's data collection time while also prioritizing measurements needs over both time and space.

Determining Paths Traversed Using Motion Trajectories Mobile devices like cell phones lack powerful exteroceptive sensors that robot use to continuously and accurately localize but their ubiquitous presence makes it desirable to use them for data collection. We apply map matching to indoor environments and show the ability to robustly to recover paths traversing from motion trajectories.

Using Fine-Grain Sensor Maps Intermittent wireless connectivity issues plagued our robot due to poor wireless handoff decisions that are used in most mobile devices today. These problems are not surprising given that robots push the boundary of mobile wireless usage. We introduce map-based handoff algorithms that resolve these wireless issues by effectively using wireless maps.

8.2 Future Work

In this thesis, we addressed several key challenges in using mobile platforms as data collectors. We explored topics like active navigation based on measurement needs and utilizing mobile platforms like cell phones equipped with less capable sensors. Ideally, we would like these data collection efforts to seamlessly gather measurements across all potential data sources in our environments, automatically analyze measurements to summarize trends and highlight anomalies, intelligently control mobile platforms for collecting measurements, and fully integrate usage of these sensor maps. While our efforts touch on a lot of these problems, it would be desirable to get to the point where data collection efforts are fully integrated into the operation and management of our indoor environments. Some directions for future work include:

Fusion of Heterogeneous Data Collection Sources This thesis focused on mobile platforms that collect sensor measurements while moving; however, static platforms can capture equally valuable sensor measurements as well. Developing a centralized platform that fuses and aggregates measurements across all these sensors would help to ensure even more timely and dense sensor maps.

Large Scale Sensor Measurement Analysis Extending data analysis efforts to combine measurements across heterogeneous sensors will require additional sensor calibration efforts. It would also be beneficial for data analysis algorithms to automatically extract spatio-temporal sensor trends and detect anomalies by learning trends across multiple environments. The ability to characterize the behavior of sensors across different environments would be valuable to help better understand how building structure, materials, and dynamic objects influence the behavior of our buildings.

Real-Time Updates of Sensor Measurement Needs Using Teams of Mobile Robots Directing subsequent data collection efforts to focus on sensor needs is powerful as it allows us to pro-actively resolve situations where additional data would be helpful. As we move towards using teams of mobile robots for data collection, there is an opportunity to explore strategies in distributed data collection efforts that quickly react to any environmental changes.

Inference About the Surrounding Environment There are some locations where we may not be able to directly capture sensor measurements (e.g. offices inaccessible for the robot or within walls of a building). With fine-grain sensor data, we may be able to infer the materials and structures behind these uncertain areas based on domain knowledge.

Discriminating Possible Paths Based on WiFi Conditions We showed that when given some path, we can compute a handoff plan that will minimize handoffs and optimize when to switch between access points. We can extend these efforts to decide between different possible paths depending on the desired wireless conditions. For example, one path might minimize the total number of handoffs while another might target higher minimum received signal strengths.

Applications in Using Fine-Grain Sensor Measurements We believe that algorithms that effectively use sensor data have only begun to emerge because access to up-to-date sensor maps has only recently been practical. This creates the opportunity for algorithms to use contextual information about the environment that enable simple algorithms to overcome traditionally difficult problems (like our map-based wireless handoffs). With the recent emergence of the Internet of Things (IoT), it seems likely that we have only just begun to fully appreciate the benefits of environment-specific algorithms.

Bibliography

Rolf Adams and Leanne Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647, 1994. 4.3

Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 1–6. ACM, 2010. 1, 7

Jorgen Bach Andersen, Theodore S Rappaport, and Susumu Yoshida. Propagation measurements and models for wireless communications channels. *Communications Magazine, IEEE*, 33(1): 42–49, 1995. 7.1

Josep Aulinas, Yvan R Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. In *CCIA*, pages 363–371. Citeseer, 2008. 2.2

Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 261–272. ACM, 2009. 2.2

Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000. 2.2, 3.1, 7.1

Paramvir Bahl, Jitendra Padhye, Lenin Ravindranath, Manpreet Singh, Alec Wolman, and Brian Zill. Dair: A framework for managing enterprise wireless networks using desktop infrastructure. In *HotNets IV*, 2005. 1, 2.4

Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006. 2.2

Aruna Balasubramanian, Ratul Mahajan, Arun Venkataramani, Brian Neil Levine, and John Zahorjan. Interactive wifi connectivity for moving vehicles. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 427–438. ACM, 2008. 7.2.1

Sangeetha Bangolae, Carol Bell, and Emily Qi. Performance study of fast bss transition using ieee 802.11 r. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pages 737–742. ACM, 2006. 7.2.1

beam. Life telepresent: working vicariously through the beam robot. ”http://www.theverge.com/2012/11/7/3611510/suitable-beam-robot-aims-for-bulletproof-telepresence”. URL `"http://www.theverge.com/2012/11/7/3611510/suitable-beam-robot-aims-for-bulletproof-telepresence"`. 2.1

Mohamed Benzeghiba, Renato De Mori, Olivier Deroo, Stephane Dupont, Teodora Erbes, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, et al. Automatic speech recognition and speech variability: A review. *Speech Communication*, 49(10):763–786, 2007. 2.3

Joydeep Biswas, Brian Coltin, and Manuela Veloso. Corrective gradient refinement for mobile robot localization. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 73–78. IEEE, 2011. 2.2

Yi-Chao Chen, Ji-Rung Chiang, Hao-hua Chu, Polly Huang, and Arvin Wen Tsui. Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 118–125. ACM, 2005. 2.2

Yin Chen, Dimitrios Lymberopoulos, Jie Liu, and Bodhi Priyantha. Fm-based indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 169–182. ACM, 2012. 2.2, 2.3

Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245. ACM, 2005. 2.2

K. Chintalapudi, A. Padmanabha Iyer, and V.N. Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2010. 2.2

Ionut Constandache, Romit Roy Choudhury, and Injong Rhee. Towards mobile phone localization without war-driving. In *Infocom, 2010 proceedings ieee*, pages 1–9. IEEE, 2010. 2.2

Nakju Doh, Howie Choset, and Wan Kyun Chung. Accurate relative localization using odometry. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1606–1612. IEEE, 2003. 2.2

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009. 2.2

Stephen Drew and Ben Liang. Mobility-aware web prefetching over heterogeneous wireless networks. In *Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on*, volume 1, pages 687–691. IEEE, 2004. 7.2.1

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006. 2.2

Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: vehicular content delivery using wifi. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 199–210. ACM, 2008. 7.2.1

Brian Ferris, Dieter Fox, and Neil D Lawrence. Wifi-slam using gaussian process latent variable models. In *IJCAI*, volume 7, pages 2480–2485, 2007. 4.1

R Fish, M Flickinger, and J Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *IEEE INFOCOM*, 2006. 3.1.1

Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3):143–166, 2003. 2.3

Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999:343–349, 1999. 2.2

D. Gusenbauer, C. Isert, and J. Krösche. Self-contained indoor positioning on off-the-shelf mobile devices. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–9, Sept 2010. doi: 10.1109/IPIN.2010.5646681. 6.2.2

Magdy F Iskander and Zhengqing Yun. Propagation prediction models for wireless communication systems. *Microwave Theory and Techniques, IEEE Transactions on*, 50(3):662–673, 2002. 7.1

Patrick Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations research*, 36(6):929–936, 1988. 5.1.2

Umar Javed, Dongsu Han, Ramon Caceres, Jeffrey Pang, Srinivasan Seshan, and Alexander Varshavsky. Predicting handoffs in 3g networks. In *Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*, page 8. ACM, 2011. 7.2.1

Marisa G Kantor and Moshe B Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, pages 629–635, 1992. 5.2.1, 5.2.2

Alonzo Kelly. Fast and easy systematic and stochastic odometry calibration. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3188–3194. IEEE, 2004. 2.2

Thomas Kollar, Mehdi Samadi, and Manuela Veloso. Enabling robots to find and fetch objects by querying the web. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1217–1218. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 2.1

Hyuk Lim, Lu-Chuan Kung, Jennifer C Hou, and Haiyun Luo. Zero-configuration, robust indoor localization: Theory and experimentation. 2005. 2.2

Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. Push the limit of wifi based localization for smartphones. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 305–316. ACM, 2012. 2.2

Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the mac-level behavior of wireless networks in the wild. *ACM SIGCOMM Computer Communication Review*, 36(4):75–86, 2006. 4.4.1, 7

Ratul Mahajan, John Zahorjan, and Brian Zill. Understanding wifi-based connectivity from moving vehicles. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 321–326. ACM, 2007. 7.2.1

Stefan Michalski. The ideal climate, risk management, the ashrae chapter, proofed fluctuations, and towards a full risk analysis model. *Experts roundtable on sustainable climate management strategies*, 2007. 1, 7

R Montemanni, D Weyland, and LM Gambardella. An enhanced ant colony system for the team orienteering problem with time windows. In *Computer Science and Society (ISCCS), 2011 International Symposium on*, pages 381–384. IEEE, 2011. 5.2.1

K.P. Murphy. Switching kalman filters. *Dept. of Computer Science, University of California, Berkeley, Tech. Rep*, 1998. 2.5

Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, and Venkata N Padmanabhan. Centaur: locating devices in an office environment. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 281–292. ACM, 2012. 2.2, 2.3

Thomas Kollar1 Vittorio Perera2 Daniele Nardi and Manuela Veloso. Learning environmental knowledge from task-based human-robot dialog. June 2013. 2.1

Anthony J Nicholson and Brian D Noble. Breadcrumbs: forecasting mobile connectivity. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 46–57. ACM, 2008. 7.2.1

Sangheon Pack, Jaeyoung Choi, Taekyoung Kwon, and Yanghee Choi. Fast-handoff support in IEEE 802.11 wireless networks. *IEEE Comms Surveys and Tutorials*, 9(1-4):2–12, 2007. 7.2.1

Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008. 1, 7

Caleb Phillips, Michael Ton, Douglas Sicker, and Dirk Grunwald. Practical radio environment mapping with geostatistics. In *Dynamic Spectrum Access Networks (DYSPAN), 2012 IEEE International Symposium on*, pages 422–433. IEEE, 2012. 4.1

Ramya Raghavendra, Elizabeth M Belding, Konstantina Papagiannaki, and Kevin C Almeroth. Understanding handoffs in large ieee 802.11 wireless networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 333–338. ACM, 2007. 2.3

Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304. ACM, 2012. 2.2, 3.1.1

Krishna N Ramachandran, Elizabeth M Belding-Royer, Kevin C Almeroth, and Milind M Buddhikot. Interference-aware channel assignment in multi-radio wireless mesh networks. In *INFOCOM*, volume 6, pages 1–12, 2006. 7.1

Ishwar Ramani and Stefan Savage. Syncscan: practical fast handoff for 802.11 infrastructure networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 675–684. IEEE, 2005. 7.2.1

Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 51–62. ACM, 2006. 4.4.1

Olof Rensfelt, Frederik Hermans, Per Gunningberg, Lars-Åke Larzon, and Erik Björnemo. Repeatable experiments with mobile nodes in a relocatable wsn testbed. *The Computer Journal*, 54(12):1973–1986, 2011. 3.1.1

J Riihijarvi, Petri Mahonen, Matthias Wellens, and Martin Gordziel. Characterization and modelling of spectrum for dynamic spectrum access with spatial statistics and random fields. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1–6. IEEE, 2008. 4.1

A. Ross. Procrustes analysis. *University of South Carolina*, 2004. 2.5

Mehdi Samadi, Thomas Kollar, and Manuela M Veloso. Using the web to interactively learn to find objects. In *AAAI*, 2012. 2.1

Michael Schilde, Karl F Doerner, Richard F Hartl, and Guenter Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, 2009. 5.2.1

Souvik Sen, Božidar Radunovic, Romit Roy Choudhury, and Tom Minka. Spot localization using phy layer information. In *Proceedings of ACM MOBISYS*, 2012. 3.1.1, 3.2.3

Alberto Serra, Davide Carboni, and Valentina Marotto. Indoor pedestrian navigation system using a modern smartphone. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, MobileHCI*, volume 10, pages 397–398. Citeseer, 2010. 2.2

Srinivasan Seshan, Hari Balakrishnan, and Randy H Katz. Handoffs in cellular wireless networks: The daedalus implementation and experience. *Wireless Personal Communications*, 4 (2):141–162, 1997. 7.2.1

Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-markie: indoor pathway mapping made easy. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, pages 85–98. USENIX Association, 2013. 2.2

Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6): 403–429, 2006. 2.2

Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Susana Brandao, Tekin Mericli, and Rodrigo Ventura. Symbiotic-autonomous service robots for userrequested tasks in a multi-floor building. *Under submission*, 2012a. 6.1.4

Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Tom Kollar, Cetin Mericli, Mehdi Samadi, Susana Brandao, and Rodrigo Ventura. Cobots: Collaborative robots servicing multi-floor buildings. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5446–5447. IEEE, 2012b. 2.1

Rodrigo Ventura, Brian Coltin, and Manuela Veloso. Web-based remote assistance to overcome robot perceptual limitations. 2013. 2.1

He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210. ACM, 2012. 2.2, 3.1.1

Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 114–123. ACM, 2008. 2.2

Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005. 2.2, 3.1, 3.1.1