UNIVERSITY OF CALIFORNIA
SANTA CRUZ

SMARTPHONE-BASED PEDESTRIAN TRACKING SYSTEM FOR VISUALLY IMPAIRED PEOPLE

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Peng Ren

March 2024

The Dissertation of Peng Ren
is approved:

Professor Roberto Manduchi, Chair

Professor Chen Qian

Professor Ricardo Sanfelice

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Smartphone-Based Pedestrian Tracking System for Visually Impaired People

by

Peng Ren

Current smartphone-based localization systems, primarily designed towards sighted individuals, offer wayfinding services by tracking a user's path. However, this design overlooks the unique navigation needs of blind individuals who utilize long canes or guide dogs and have distinct movement patterns. To bridge this gap, this thesis introduces novel localization techniques tailored for blind pedestrians in both indoor and outdoor settings. These techniques avoid the need for BLE beacons and Wi-Fi, as well as camera-based systems, all of which are impractical for blind users. Instead of these options, the proposed methods utilize IMU sensors, allowing users to conveniently place their phones in their pockets without the requirement of any external infrastructure.

Indoor localization in the absence of maps is addressed in this thesis through a unique combination of a Mixture Kalman filter and a GRU-based straight walking detector. Together, these form a two-stage turn detector that operates under the assumption that corridor intersections occur at 45° or 90° angles. In situations where maps are accessible, the research incorporates two Pedestrian Dead Reckoning (PDR) methods with the map data via a particle filter. In outdoor settings, this thesis expands the use of IMU sensor data by integrating it with GPS signals through a particle filter. This method creates a flexible model effective in both open areas and in environments with wall constraints, as specified by maps. Comprehensive testing

of these systems involved trials with the WeAllWalk dataset, containing data from visually impaired walkers, and user studies conducted using two separate iPhone applications for indoor and outdoor localization. Results from these tests clearly demonstrate the effectiveness of the proposed localization solutions.

Acknowledgments

First, I would like to express my deepest gratitude to my professor, Roberto Manduchi. I am thankful for the opportunity that he provided, which allowed me to engage in unique and beneficial tasks. His guidance revealed the elegance of academia to me during my master's studies and continued throughout my Ph.D. program. Over the past six years, his generous and insightful mentorship has not only sharpened my problem-solving skills but also taught me the essentials of research methodology. His encouragement played a crucial role in helping me overcome the numerous challenges encountered during the development of effective solutions for various projects. I am deeply appreciative of his help, without which my current achievements would not have been possible.

I also extend my sincere thanks to Professor Chen Qian and Professor Ricardo Sanfelice for serving on my defense committee and reviewing my dissertation. Additionally, I am grateful to my collaborators on various projects, particularly Fatemeh Elyasi, Chia Hsuan Tsai, and Jonathan Lam. The brainstorming sessions and discussions we shared are unforgettable. Further, I express my appreciation to the other members of the UCSC Computer Vision Lab, especially Yunqian Cheng and Swati, for their indispensable assistance in numerous aspects.

My profound gratitude goes to my mother for her unwavering support, regardless of the various difficulties I faced during this demanding yet rewarding journey, and to my father for his consistent assistance, providing a solid foundation. Additionally, I am thankful to Fang Chieh Chou, my mentor at the Nissan Advanced Technology Center in Silicon Valley, where I completed a summer internship. He introduced me to the methodologies of industrial research,

and the experience was invaluable.

The following published materials are reprinted in this thesis:

1. Tsai, C. H., Elyasi, F., Ren, P., & Manduchi, R. (2024). All the Way There and Back: Inertial-Based, Phone-in-Pocket Indoor Wayfinding and Backtracking Apps for Blind Travelers. arXiv preprint arXiv:2401.08021.

2. Ren, P., Lam, J., Manduchi, R., & Mirzaei, F. (2023, October). Experiments with Route-Nav, A Wayfinding App for Blind Travelers in a Transit Hub. In Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility (pp. 1-15).

3. Ren, P., Elyasi, F., & Manduchi, R. (2021). Smartphone-based inertial odometry for blind walkers. Sensors, 21(12), 4033.

Chapter 1

Introduction

In our daily lives, the concept of travel involves physically moving from one position to another. Whether it's going from home to school, a restaurant to a movie theater, or the entrance of a hospital to a doctor's office, these travel operations, while becoming easier when we are familiar with the path to different destinations, require an initial information collection phase known as "wayfinding." Gathering the necessary data from the surrounding environment or a map during this process can be challenging for everyone. Many people, to varying degrees, have experienced the frustration of getting lost in an urban environment or trying to locate a specific room in a large building. Such experiences can lead to anxiety and exhaustion. This issue is even more severe for blind people, as it can be not only anxiety-inducing [19] but also potentially dangerous since they lack access to visual signs ([12], [65]) present in indoor or outdoor environments, such as hospitals, airports, or transit hubs. While infrastructure like tactile paving [8], [45] and acoustic wayfinding, such as soundscape design [25], has been developed to aid visually impaired individuals, the task of wayfinding for them still presents

significant challenges.

In the context of wayfinding, one of the most crucial objectives is to determine the user's current location, as navigation would be impossible without this information. The widespread use and portability of smartphones make them ideal devices for providing indoor and outdoor localization services, made feasible by the advancements in hardware. Localization relies primarily on the various sensors within the smartphone and prior map data. It's worth noting that map-less wayfinding techniques are also available ([23, 42, 82]).

Numerous studies have been conducted in this field, addressing both outdoor and indoor navigation challenges. In outdoor scenarios, GPS is widely used; however, it can be unreliable when satellite signals are reflected or partially blocked. Indoor environments, on the other hand, present challenges for GPS due to signal blockage and multipath issues. To address indoor navigation, several technologies have been explored, including Bluetooth Low Energy (BLE) beacons [83], Wi-Fi [9], magnetic field [33], and Ultra-Wideband (UWB) [3]. Each of these methods has its own limitations.

BLE beacons and Wi-Fi require environment fingerprinting and ongoing maintenance, which can be time-consuming and costly. The installation of BLE beacons is essential for the operation of this system. Magnetic field-based localization also relies on fingerprinting to map magnetic patterns to a user's location. UWB, while capable of providing accurate indoor localization, requires external infrastructure support and specialized hardware in smartphones, which may not be available in all devices.

Alternatively, methods that don't rely on external infrastructure exist. Visual-based odometry, for instance, utilizes the smartphone's camera. However, this approach necessitates

an unobstructed field of view for the camera. It may not gather sufficient visual data if the phone is placed in a pocket or if the camera only captures images of the floor, ceiling or crowd, which may not include useful features.

In comparison to the methods mentioned above, inertial navigation systems offer several advantages. They can operate without relying on external infrastructure, making them suitable across different scenarios. Inertial sensors are also power-efficient, and the system can even function when the smartphone is placed in the user's pocket. However, inertial navigation has its downsides, including errors that slowly add up over time, leading to incorrect position information.

To address the drift issue, researchers have proposed various solutions. For instance, when an inertial measurement unit (IMU) is attached to a user's foot, Zero-velocity updates (ZUPT) and Zero Angular Rate Update (ZARU) [59] can be employed to mitigate drift. Utilizing Heuristic Heading Reduction (HDR) [10] represents an alternative approach to constraining drift. Combining inertial sensors with other technologies like Wi-Fi localization, which is drift-resistant, can also enhance robustness. If an environmental map is available, it can be used to apply constraints on the reconstructed path. Machine learning techniques can also be utilized to achieve more accurate results.

It's worth noting that many existing inertial sensor navigation systems are designed for sighted individuals and may not be suitable for blind walkers. Gait differences between sighted people and visually impaired individuals who use long canes or guide dogs can affect heading stability, leading to challenges in navigation accuracy. Blind travelers may also stop or reorient themselves when encountering obstacles or people, introducing errors into conven-

tional navigation systems. Therefore, developing a system tailored to individuals with visual impairments requires using datasets collected from blind walkers rather than sighted ones. The importance of the data source becomes even more critical when employing data-driven machine learning methods that rely heavily on training data. In this context, the WeAllWalk dataset, created by Professor Manduchi's group, stands out as the only publicly available dataset collected from individuals with visual impairments, which has been extensively utilized in various models introduced in this thesis.

This thesis centers on smartphone-based localization techniques designed to assist blind individuals in both indoor and outdoor settings. Specifically, we have chosen the iPhone as our experimental platform due to its popularity within the blind community [50]. In the indoor context, our research is concentrated on several buildings within the UCSC campus. For the outdoor scenario, our focus is on the Palo Alto transit hub, which encompasses both open ground areas and underground tunnels.

In the indoor environment, we rely on IMU sensors, including accelerometers and gyroscopes, in conjunction with prior map information. These data sources are integrated using a sensor fusion algorithm, specifically the particle filter. We also delve into the map-less scenario within indoor settings, where we employ a Kalman filter-based model to determine the user's heading while following to predefined constraints.

In the case of the transit hub, we leverage GPS signals to enhance the performance of the particle filter, especially in outdoor open spaces. Altimeter data is employed to facilitate the transition between the overground and underground maps. It is important to note that the localization systems discussed here can be easily extended to other buildings or transit hub

centers.

The thesis is organized as follows:

- Chapter 2 focuses on the related work, with a particular emphasis on the various sensors mentioned earlier and IMU-based localization methods applicable in both indoor and outdoor scenarios.

- Chapter 3 presents a two-stage turn detection algorithm based on the Mixture Kalman Filter designed to operate in map-less situations.

- In Chapter 4, we delve into the fundamental concept of the particle filter and introduce the Mean-Shift algorithm, which can be employed for automatic particle clustering.

- Chapter 5 and Chapter 6 provide a comprehensive discussion of the application of the particle filter in indoor and outdoor settings, emphasizing distinct steps and approaches for handling extreme cases.

- Chapter 7 offers an evaluation of the two-stage turn detection model and examines the paths reconstructed through various methodologies using the Weallwalk dataset.

- Chapter 8 and Chapter 9 offer an extensive exploration of two navigation applications: one designed for indoor use and the other for transit-hub scenarios. These chapters encompass model details, user interface design, and experimental results, along with solutions to the challenges encountered.

- Chapter 10 conducts an ablation study on different components of the localization model in the transit-hub scenario, confirming the efficiency of integrating data from various

sources.

- Finally, Chapter 11 serves as a conclusion, summarizing the entire thesis and highlighting open problems for future research.

Chapter 2

Related Work

2.1 Indoor/Outdoor Localization Systems

In this section we talk about different sensors that are used in localizing a smartphone user in either indoor or outdoor scenarios. We begin with indoor localization systems, which includes sensors: Wi-Fi, BLE-Beacon, Camera, UWB, and Magnetic field. Then we focus on the GNSS signal which is widely used in the outdoor scenario. Notice that, the inertial sensor which is extensively used in this thesis, are discussed in a separate section right after this one.

2.1.1 Wi-Fi-Based Indoor Positioning

Wi-Fi positioning utilizes the signal strength received from multiple Wi-Fi beacons to ascertain the precise location of an individual within a given environment. To achieve this, an initial fingerprinting operation must be conducted, during which Wi-Fi signals are collected from various locations within the environment. The strength of these received signals serves

as a unique fingerprint for each location. Consequently, the first step involves constructing a fingerprint database that correlates vectors of received signal power with known locations. Subsequently, during real-time operation, the user's location can be estimated based on the received signal power vector [72][77]. It is important to note that in many public spaces, Wi-Fi beacons may already be installed for other purposes, eliminating the need for additional infrastructure. However, if any of these beacons are deactivated, the fingerprint data may require recalibration. Furthermore, the accuracy of fingerprinting can be affected by minor environmental changes, such as the rearrangement of furniture, potentially rendering Wi-Fi positioning less reliable [70]. Additionally, it is worth mentioning that obtaining accurate information about the walker's position during data collection for database construction can be a challenging task.

2.1.2 BLE-Beacon-Based Indoor Positioning

A comparable approach to Wi-Fi-based localization is offered by BLE (Bluetooth Low Energy) positioning. In this scenario, BLE beacons prove cost-effective and durable over long periods. Presently, two primary Bluetooth Low Energy (BLE) beacon-based positioning methods are available [84]: the range-based method and the fingerprinting-based method. In the former method, estimates of distances between the user and various BLE-beacon stations are calculated. If the precise positions of the BLE-beacon stations are known, the user's position can be determined [73]. However, it is important to note that the range-based method generally exhibits limited effectiveness indoors. The second method mirrors the approach discussed earlier in the Wi-Fi section. This involves initially constructing a fingerprinting database during the offline phase. Subsequently, in the online phase, newly acquired fingerprints are compared

against the entire database to ascertain the user's location. BLE beacons have demonstrated successful use in the localization of visually impaired individuals within indoor environments [2]. Nonetheless, it is crucial to acknowledge that this system demands supplementary infrastructure and support. Furthermore, similar to the Wi-Fi-based approach, the development of a database comprising received power vectors at known locations can be challenging due to difficulties in accurately determining the location during data collection [83].

2.1.3 Visual Odometry

Sighted individuals acquire crucial spatial information through their sense of sight. Consequently, utilizing visual information for indoor localization holds significant promise. For instance, visual odometry relies on well-established techniques such as Structure from Motion or visual SLAM [61]. This approach enables the reconstruction of camera location, orientation, and world structure up to scale by tracking visual features from one frame to another [61]. Scale information can be derived through the utilization of data from acceleration sensors or depth sensors. Notably, the Apple ARKit framework [1] seamlessly integrates visual odometry with inertial odometry to provide camera position and orientation within a global coordinate framework. Visual odometry has also previously demonstrated utility in aiding visually impaired individuals [26]. Nevertheless, it is important to emphasize that continuous, unobstructed camera views of the environment are essential for visual odometry to function effectively. Consequently, visual odometry may encounter limitations when the camera's view is obstructed by people or when the user keeps the camera in their pocket. Considering that

[1] https://developer.apple.com/augmented-reality/

visually impaired users typically require one hand for holding a walking cane or guiding a dog, the visual odometry-based method may prove impractical for their needs.

2.1.4 Magnetic-Based Indoor Positioning

The indoor magnetic field, vulnerable to persistent distortions caused by electrical appliances or steel frame structures in buildings, can serve as a valuable resource for localization. A fingerprinting approach, similar to those discussed earlier, is necessary in this context, with measured values comprising the magnitude of the magnetic field [31]. Researchers have successfully employed this technology for indoor user localization using smartphones [2]. However, it is important to note that the construction of a magnetic map can be a time-consuming process, and the magnetic field has been observed to be variable in time.

2.1.5 Ultra-Wide Band-based Indoor Localization System

Ultra-Wide Band (UWB), widely recognized as one of the most common building-dependent localization technologies [54], offers precise indoor user localization capabilities. This method relies on two distinct sets of hardware: the UWB tag, which is attached to the user's body, and the anchors positioned throughout the environment, with their precise locations predetermined. During operation, the anchors and the UWB tag engage in communication through UWB signals, employing the Time of Arrival (TOA) algorithm [51] to measure the distances between the anchors and the tag. Subsequently, the trilateration algorithm is employed to determine the user's precise position. Alternatively, other techniques like Time Difference of

[2] https://www.indooratlas.com/

Arrival (TDOA) [17] or Angle of Arrival (AOA) [46] can be employed for the same purpose. An inherent advantage of the UWB method lies in its immunity to multipath interference [6].

However, there are certain drawbacks associated with the UWB method. It necessitates the pre-installation of external infrastructures, and users are required to carry specific UWB tags. While UWB technology is becoming increasingly accessible in modern smartphones, such as the iPhone 11 models and later [3], it may not be available in slightly older devices, thus limiting its widespread usage.

2.1.6 GNSS-Based Outdoor Localization Systems

GPS, an abbreviation for Global Positioning System, is a technology that can provide users in outdoor scenarios with their absolute positions when they have the necessary GPS equipment. Nowadays, affordable GPS technology is embedded in smartphones [38] and smartwatches [71], making it a cost-effective yet highly efficient solution for users interested in determining their outdoor locations. It finds application in a wide range of uses, including navigation, recommendations for nearby restaurants, route tracking, and so on.

The fundamental concept of GPS[78] involves multiple satellites orbiting the Earth, where GPS receivers receive signals from these satellites, enabling the calculation of the distance between the user's receiver and each satellite. By combining these distances, the user's current position on Earth can be determined. GPS, although commonly associated with a specific technique, originally referred to the positioning service provided by the USA. Other countries also offer their satellite positioning services; for instance, the European Union provides

[3] https://support.apple.com/en-us/109512

Galileo[52], Russia offers Glonass [60], and China provides the Beidou service[81]. When users employ satellites from multiple sources mentioned above, the technique is commonly referred to as GNSS (Global Navigation Satellite System).

While GNSS services can combine signals from numerous satellites worldwide, it is essential to note that the achievable localization accuracy can be limited. This limitation arises due to various errors introduced during signal transmission between satellites and the user's equipment [36]. These errors include satellite clock inaccuracies, satellite position discrepancies, errors induced by the ionosphere and atmosphere, and the well-known multipath problem [66], which arises from signal reflections off tall buildings, trees or ground. These errors often make the measured distance between the equipment and remote satellites inaccurate. In fact, the raw distance calculated in this manner is termed "pseudorange," indicating its high unreliability. Techniques aimed at mitigating these inaccuracies have been extensively studied within a specific research domain, including methods such as Differential GPS [49], RTK-GPS (Real-Time Kinematic Global Positioning System) [43], and PPP (Precise Point Positioning) [40]. However, discussing the advantages and drawbacks of these methods falls beyond the scope of this thesis. Nevertheless, a valuable takeaway is that, irrespective of the efforts invested, relying solely on GPS for high-precision localization services in environments such as urban can remain quite challenging, particularly in scenarios where GPS signals are not consistently available or when satellite positions in the sky closely overlap.

2.2 Inertial Sensor-Based Localization

The inertial sensors, including accelerometers and gyroscopes, integrated into a smartphone can serve as a valuable tool for odometry. This approach offers the advantage of not requiring external infrastructure and functions effectively even when the phone is placed in a user's pocket. Furthermore, inertial sensors consume less power compared to a camera. However, a significant drawback of the IMU (Inertial Measurement Unit) sensor is its susceptibility to drift. Both accelerometers and gyroscopes within the IMU are prone to various noise sources, such as constant bias error, temperature effects, random walk noise, and flicker noise [75]. These noise sources are responsible for the drift errors that occur in various methods utilizing the IMU sensor, all of which are discussed below.

2.2.1 Strap-Down Inertial Navigation

The strapdown inertial navigation algorithm utilizes data from accelerometers and gyroscopes to reconstruct a user's trajectory [22]. The concept involves integrating data from the gyroscope to estimate the phone's attitude relative to an initial frame of reference, defined with the Z-axis pointing downward (in the direction of gravity). At each time step, readings from the accelerometer and gyroscope are reoriented to the initial reference frame. Subsequently, after double-integrating the user's acceleration with gravity removed and obtaining azimuth information from gyro integration, these two pieces of information are combined to determine the current position. Unfortunately, low-cost inertial sensor readings can be affected by bias and noise mentioned above, which, when integrated, lead to directional and positional inaccuracies.

2.2.2 Pedestrian Dead Reckoning

Pedestrian Dead Reckoning (PDR) [76] presents a straightforward approach that eliminates the need for double integration of acceleration. To calculate the distance traveled, a PDR system detects steps and estimates step length. The user's facing direction (azimuth) can be obtained through the integration of data from the gyroscope (note that this information is readily provided by modern operating systems). Step detection can be accomplished using data from accelerometers and/or gyroscopes. For instance, Alzantot et al. [4] used a finite state machine and accelerometer data magnitude to perform step detection. To accurately estimate step length, various methods have been proposed, relying on predefined coefficients and user data such as accelerometer readings, user height, and step frequency [76]. More recently, deep learning methods have been applied for step length estimation [32].

It is important to note that even when step counts and step lengths are accurately estimated, the estimated azimuth may suffer from drift due to any remaining bias in the gyroscopes, leading to substantial positioning errors over time. Furthermore, both PDR and strapdown integration methods measure the smartphone's orientation rather than the user's orientation. Therefore, these systems may fail if the user repositions the phone while walking. Methods employing Principal Component Analysis (PCA) have been introduced to address this issue and ascertain the user's movement direction [41][55].

2.2.3 Learning-Based Odometry

With the availability of larger sensor datasets for training, machine learning-based odometry techniques have emerged. Among these methods, Chen et al. [13] employed a seg-

mentation approach to divide inertial data into windows and trained a deep learning model to estimate velocity and orientation within each window. The model introduced in [80] utilizes data from accelerometers and gyroscopes to regress walking velocity vectors. Yan et al. [79] introduced RoNIN, a position estimation model that achieves state-of-the-art results when applied to data from sighted individuals. Taking raw accelerometer data, raw gyroscope data, and attitude data as inputs, RoNIN generates velocity vectors that can be integrated to determine user positions. One notable feature of RoNIN is its ability to measure user orientation rather than relying on phone orientation. Additionally, the authors have ensured RoNIN's compatibility with various phone attitudes through a process known as coordinate frame normalization. During this transformation, sensor data is converted into a coordinate frame where the Z-axis points toward the ground. It's important to note that while RoNIN represents a form of dead reckoning (using data from accelerometers and gyroscopes), its structure requires 200-length data as input. The design with a limited window size also presents challenges in accurately mitigating drift, and RoNIN still encounters drift-related issues.

2.3 Localization for Blind People

This section explores navigation systems designed for visually impaired individuals, focusing on two distinct environments: indoor and outdoor.

2.3.1 Indoor Scenario

Indoor navigation systems can offer numerous benefits to blind travelers, enhancing various aspects of their mobility. Such systems can encourage them to travel, assist in locating specific destinations within buildings, and provide guidance for retracing their steps back to the starting point. Several studies focused on wayfinding for visually impaired individuals have made notable contributions. Fusco et al. [27] combined the Visual-Inertial Odometry algorithm with a particle filter to leverage map constraints and track visually impaired individuals. Additionally, computer vision technology is employed to recognize exit signs, reducing localization uncertainty. Leung et al. [44] designed a head-mounted stereo system to aid blind individuals in navigating through obstacles. Leveraging a BLE beacons network, the system introduced in [2] effectively localizes visually impaired users. Riehle et al. [57] explored the use of magnetic sensing to address indoor navigation challenges for the visually impaired. Furthermore, Fallah et al. [21] and Riehle et al. [58] conducted experiments with inertial odometry systems for blind navigation. Flores and Manduchi [23] introduced an indoor backtracking system based on step counts and turn detection.

2.3.2 Outdoor Scenario

Outdoor navigation systems are incredibly valuable for blind travelers, although they present more challenges compared to indoor scenarios due to the complexity and size of outdoor environments. Enabling blind travelers to navigate outdoors can greatly increase their mobility and improve their overall quality of life. Similar to indoor navigation, extensive research has been conducted in this domain. [28] and [29] proposed the use of GPS for personal naviga-

tion systems tailored to blind travelers. Nowadays, commercially accessible navigation systems like Apple Maps and Google Maps utilize GPS for positioning and employ geographical information datasets for guidance. Additionally, specialized apps designed for visually impaired individuals, such as GoodMaps Outdoors, APH Nearby Explorer, and BlindSquare, are readily available. Recognizing that GPS localization can be prone to inaccuracies, both Apple Maps and Google Maps have implemented modes that enhance localization accuracy by utilizing the camera to capture images of buildings, which are then matched against 3-D point datasets when available. [20] created a system that uses a single camera worn on the body and visual SLAM technology to deliver highly accurate global positioning and navigation. [63] uses advanced deep learning methods to analyze photos taken with smartphones for identifying obstacles and measuring distances. [47] introduced a support system that utilizes BLE beacons on buses and stops, making it easier for visually impaired people to locate public transport and stops, thereby improving their travel experience. [48] merges GPS with BLE beacons to overcome GPS limitations and enhance accuracy in determining one's location.

2.4 Inertial Sensor Data Sets

To train machine learning models effectively, well-calibrated datasets with accurate labels are essential. Such datasets also serve as valuable resources for method evaluation and comparisons. In the field of inertial sensors, numerous datasets are available today. For instance, the dataset introduced in [80] comprises IMU sensor data collected from ten sighted individuals, with the smartphone placed in four different positions on their bodies. An integrated visual-

inertial odometry system is used to obtain precise ground truth. Chen et al. [14] have provided a dataset containing inertial and magnetic data gathered from a variety of phones, users, and motion scenarios, reflecting phone movements in diverse situations. They utilize an Optical Motion Capturing System to provide highly accurate ground truth labels. The dataset proposed in [79] encompasses inertial sensor data collected from 100 participants, allowing unrestricted device placement. A 3D tracking phone is employed to achieve high-quality ground truth.

Unfortunately, the datasets mentioned earlier only contain data from sighted individuals, making them unsuitable for developing models for blind travelers. The WeAllWalk dataset[4] [24] is the sole publicly accessible dataset gathered from visually impaired individuals. It encompasses inertial data recorded at a 25Hz frequency, collected from ten blind travelers utilizing either a walking cane or a guide dog. Specifically, nine participants utilize walking canes, while three choose guide dogs. Additionally, the dataset includes data from five sighted individuals. The WeAllWalk participants collectively covered a total distance of 7 miles during their walks, each carrying two iPhones placed at different locations on their bodies. This dataset records inertial sensor data, attitude data, and timestamps for specific waypoints, typically located at the endpoints of straight corridor segments. Furthermore, it captures instances when a blind traveler encountered obstacles, opened doors, or came to a halt, all categorized as ”features” within the dataset.

[4] https://datadryad.org/stash/dataset/doi:10.7291/D17P46

2.5 Conclusion

In this section, we delve into the related work relevant to this thesis. Initially, we investigate the drawbacks associated with indoor localization systems based on Wi-Fi, BLE-Beacon, visual odometry, UWB, and magnetic field technologies. Subsequently, we shift our focus to the limitations of the GNSS system, primarily suited for outdoor scenarios. Following that, we explore the advantages and disadvantages of inertial sensor-based localization and initiate a discussion on various methods to utilize IMU sensor data, including strap-down navigation, pedestrian dead reckoning, and learning-based odometry. Our attention then turns to the challenges faced by blind individuals, and we discuss the diverse localization techniques employed to facilitate their navigation in both indoor and outdoor settings. Lastly, we examine the availability of inertial sensor datasets collected for both sighted and blind individuals.

Chapter 3

Map-Less Indoor Turn Detection

In this section, we explore how to perform turn detection while people are inside a building without using a map. First, we'll describe the problem. Then we'll discuss the specific Kalman filter used, as well as a simplified version that's easier to implement. After that, we'll introduce the Mixture Kalman Filter model [15] and explain the two-stage process for detecting turns, which helps make our method more precise.

3.1 Problem Statement

Tracking blind people indoors using only smartphone IMU data is a complex task. As previously discussed, we're avoiding the use of cameras or external infrastructure like BLE beacons or Wi-Fi and instead rely on methods such as Pedestrian Dead Reckoning (PDR). For successful tracking, we must count steps, measure stride length, and accurately determine the user's orientation. While we can count steps and measure stride length, ensuring the correct orientation without visual hints is particularly challenging. We determine orientation using

the azimuth angle, which represents the facing direction. This azimuth data is calculated by measuring the angle between the phone's Y-axis and the world's Y-axis after projecting the phone onto a plane—a horizontal surface defined by the X and Y axes of the world frame. However, the azimuth can drift over time due to the inherent inaccuracies of the IMU sensor, and this drift can render the orientation data unreliable after just a short period. Thus, the primary challenge is to correct this azimuth drift. We will next explore an effective model, the Mixture Kalman Filter, that addresses this issue.

3.2 Kalman Filter - Review

Before we delve into the Mixture Kalman Filter model, it's important to understand the basic Kalman filter [74]. The Kalman filter is a Bayesian filter known for its efficiency in linear problem-solving. Although the Kalman filter we use here is quite basic, the rationale and intuition behind the equations are valuable to discuss. The objective of the Kalman filter in this context is to monitor the drift in the azimuth data. This data can be represented as:

$$z_t = O^*(t) + d_t^* \tag{3.1}$$

where z_t is the measured azimuth at time t, $O^*(t)$ is the actual orientation of the user at time t, and d_t^* is the actual drift noise at time t. One can determine the value of $O^*(t)$ if the drift d_t^* is accurately tracked. However, without access to a map, tracking drift for different visually impaired users—who may each experience unique drift values—becomes exceedingly difficult. To manage this, we begin with a fundamental assumption:

As users walk through corridors, their orientation is typically a multiple of 90 degrees

or 45 degrees, a pattern consistent with most indoor building layouts. Mathematically, the orientation can be expressed as:

$$O^k(t) = \frac{k \times 360^\circ}{N} \tag{3.2}$$

where N equals 4 or 8, and k varies from 1 to N.

The Kalman filter model leverages this 4-class or 8-class orientation hypothesis by utilizing a specially designed process. This involves establishing a transition matrix that captures the likelihood of each potential turn. The transition probability is outlined as:

$$P(O^k(t)|O^m(t-1)) = \begin{cases} q, & \text{if } m = k \\ (1-q)/2, & \text{if } |m-k| = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Here, $O^k(t)$ and $O^m(t-1)$ represent the orientation candidates at the current and previous time steps, respectively. Transition probabilities default to zero if the absolute difference between m and k exceeds 1, indicating that turns larger than 360°/N between two timestamps are not feasible (e.g., an 180° turn is treated as two consecutive 90° turns for $N = 4$). This assumption simplifies parameter selection. The value of q may vary based on the user's walking pattern, with a higher q during non-straight walking intervals when orientation changes are more probable. With these assumptions in place, we will now examine the Kalman filter equations in detail.

3.2.1 State Initialization and Matrix Definitions

Standard matrices are initialized as follows:

State: The initial step in configuring a Kalman filter is to define its states, which comprise the data that the filter is designed to estimate. Our model aims to monitor drift and determine orientation for each Kalman filter, and is represented by the state vector:

$$x = \begin{bmatrix} d \\ 1 \end{bmatrix} \tag{3.4}$$

Here, d represents the drift within azimuth data.

Observation Matrix: Let $O(t)$ denote the orientation of the Kalman filter at time t. The observation matrix H is then defined as:

$$H = \begin{bmatrix} 1 & O(t) \end{bmatrix} \tag{3.5}$$

The observation matrix transforms states from the state space to the observation space. Multiplying H by x will reconstruct the azimuth data, according to the aforementioned model.

Covariance Matrix: The covariance matrix estimates the covariance between states and the variance of each state individually. We focus solely on the variance of drift, located at the top-left corner of the covariance matrix. The variance/covariance of $O(t)$ is considered to be zero since its value is sampled from a probability distribution, which is detailed later.

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.6}$$

State Transition Matrix: For the prediction step, the state transition matrix F remains an identity matrix, indicating no change to the state.

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

Process Noise: The matrix Q represents the internal noise of the Kalman filter model. In our model, we account for the process noise associated with the drift state as Gaussian noise with a standard deviation of σ_w, and we assume that there is no noise in the other states.

$$Q = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.8}$$

Observation Noise: Observation noise, distinct from model noise, is modeled as Gaussian noise with a standard deviation of σ_v.

$$R = \sigma_v^2 \tag{3.9}$$

After defining and initializing the matrices, the Kalman filter begins with the prediction step.

3.2.2 Prediction Step

Without new observations, the model predicts the next state using the current state:

$$x_{t|t-1} = F \cdot x_{t-1} \tag{3.10}$$

Given that F is an identity matrix, there is no change in the state.

The covariance matrix P is also predicted:

$$P_{t|t-1} = F \cdot P_{t-1} \cdot F^T + Q \tag{3.11}$$

It is essential to ensure that $P_{t|t-1}$ retains zeros except for the top-left element, which represents the drift variance.

Using the previous orientation $O^m(t-1)$ from the Kalman filter, we can calculate the likelihood of each possible orientation $O^k(t)$ using the transition matrix.

$$P(O^k(t)|O^m(t-1)), \quad k = 1,\ldots,N \tag{3.12}$$

3.2.3 Update Step

During this step, the azimuth observation refines the state estimate. The drift d is modeled as a Gaussian distribution, allowing for the calculation of the likelihood of observing z_t given the drift model and a hypothesized orientation $O^k(t)$:

$$P(z_t|d_{t-1}, O^k(t)), \quad k = 1,\ldots,N \tag{3.13}$$

The variance of the drift error σ_d^2 is estimated using the top-left element of the P matrix. This variance is utilized to evaluate the observed drift against the Kalman filter's drift model by computing the probability density function (PDF):

$$P(z_t|d_{t-1}, O^k(t)) = \mathcal{N}(Diff(z_t, O^k(t) + d_{t-1}); 0, \sigma_d^2), \quad k = 1,\ldots,N \tag{3.14}$$

To ensure precise calculation of the angular difference, we must take into account the cyclical characteristic of angles. For example, the angular difference between 0 degrees and 90 degrees should be correctly computed as 90 degrees, and not the supplementary angle of 270 degrees. The correct angular difference is determined by a specific formula that respects the circular continuity of angles:

$$Diff(z_t, \Lambda_t) = ((z_t - \Lambda_t) + \pi)\,\%2\pi - \pi \tag{3.15}$$

After computing $P(z_t|d_{t-1}, O^k(t))$, it can be integrated with the probability of different possible orientations.

$$P(z_t|d_{t-1}, O^k(t)) \cdot P(O^k(t)|O^m(t-1)), \quad k = 1, \ldots, N \tag{3.16}$$

By accumulating the likelihood across all orientations, we can calculate $P(z_t|KF)$, the likelihood of observing the current azimuth value z_t given the Kalman filter. This likelihood can be interpreted as the significance of the current Kalman filter. This measurement will subsequently be utilized in the Mixture Kalman Filter model to update the Kalman filter weight. Furthermore, to determine the new orientation $O(t)$ of the Kalman filter, we can sample it:

$$O(t) \propto P(z_t|d_{t-1}, O^k(t)) \cdot P(O^k(t)|O^m(t-1)), \quad k = 1, \ldots, N \tag{3.17}$$

Sampling from this resultant distribution allows for the assignment of $O(t)$ at each time step. This process forms a continuous orientation sequence within the Kalman filter:

Upon selecting $O(t)$, the observation matrix H is updated, and the innovation, the discrepancy between the observed and estimated drift, is computed as:

$$z_t - H \cdot x_{t|t-1} = Diff(z_t, O(t) + d_{t-1}) \tag{3.18}$$

Following this process, the innovation derived from measurements is applied to improve the state estimate. It is important to note that within the state vector, only the drift component needs to be updated. The other components of the state vector are reset to their initial values after this update step.

$$x_t = x_{t|t-1} + K_t \cdot (z_t - H \cdot x_{t|t-1}) \tag{3.19}$$

The Kalman gain K_t, calculated using R and P, measures the relative reliability of the observations versus the model predictions. It is derived as follows, where the numerator represents

the predicted system uncertainty transformed to the observation space, and the denominator accounts for observation noise. Note that the results are subsequently transformed back into the state space:

$$K_t = H^{-1} \cdot \frac{H \cdot P_{t|t-1} \cdot H^T}{H \cdot P_{t|t-1} \cdot H^T + R} \tag{3.20}$$

The Kalman gain influences the degree to which the predictions are adjusted based on new observations. A smaller K_t suggests a higher level of trust in the model's predictions over the measurements, whereas a larger K_t suggests higher trust in the observational data. Finally, the covariance matrix P is updated to reflect the revised uncertainty after incorporating the observation:

$$P_t = (I - K_t \cdot H) \cdot P_{t|t-1} \tag{3.21}$$

Given that the orientation is not tracked via the covariance matrix P, we can simplify the update process. Only the top-left element of P and the first element of the state vector x are updated after each iteration:

$$P_t[0,0] = \frac{1}{\frac{1}{\sigma_v^2} + \frac{1}{P_{t|t-1}[0,0] + \sigma_w^2}} \tag{3.22}$$

$$x_t[0] = x_{t|t-1}[0] + \frac{P_t[0,0]}{\sigma_v^2} \cdot (z_t - H \cdot x_{t|t-1}) \tag{3.23}$$

3.3 Mixture Kalman Filter

The Kalman filter described above, also referred to as the Conditional Dynamic Linear Model (CDLM) [15], relies on an indicator variable O(t) to track the system's drift via a standard Kalman filter approach. To refine the estimation of the target mixed Gaussian distri-

bution $P(State|Observation)$, a Mixture Kalman Filter (MKF) [15] is employed to approximate this distribution and offer enhanced accuracy.

MKF algorithm is an algorithm that mixes different kalman filter to obtain a better results. The system makes decisions by considering the outputs from all the Kalman filters, each assigned a weight that reflects its agreement with the observed data, taking into account various potential orientations. The MKF algorithm is shown in Algorithm 1.

Algorithm 1 Update Kalman Filters with New Observations

1: **Initialization:** Each Kalman filter begins with equal weights, $\frac{1}{\text{number of Kalman filters}}$.
2: **New Observation:** Upon receiving a new observation, the system uses the SW detector to determine whether the user's walking status is normal.
3: **if** user is walking normally **then**
4: Use a specific set of turn probabilities for normal walking.
5: **else**
6: Use a different set of higher turn probabilities.
7: **end if**
8: Update each Kalman filter based on the observation and the chosen turn probabilities.
9: Normalize the weights of each Kalman filter.
10: Remove any Kalman filter whose weight is less than 10^{-12}.
11: Initiate a resampling process if the Coefficient of Variation for all Kalman filters exceeds a predefined threshold.
12: Normalize the weights of each Kalman filter again.

At the beginning, each kalman filter is equally weighted, since each of them are the

same. Upon receiving new azimuth data, we first check whether the user is walking normally. The turn probabilities are determined based on the user's walking condition. Following the update of the Kalman filter as we discussed above, we do the kalman filter weights normalization to make sure the sum of weights equals 1, in this case one can view the weight of each kalman filter as its probability, which is convenient. Subsequently, we perform the resmapling process. A metric named COV is utilized to measure the discrepancy of weights of each kalman filter.

$$\text{COV} = \frac{\sigma}{\mu} \tag{3.24}$$

σ is the standard deviation of the weight list, and μ is the mean of the weight list. The goal here is to provide a list of kalman filters whose weights are almost equally distributed. If the discrepancy is too large, it means that some kalman filter with low weights needs to be removed. The COV threshold is set to 0.3 in this case. A standard resampling or rejuvenation process is then applied to maintain an effective list of Kalman filters. During resampling, filters are selected proportional to their weights. Once resampling is done, we do the weights normalization again. Finally, the final output is performed by considering the normalized weights of each kalman filter.

For this study, a set of 50 Kalman filters is utilized. The parameters σ_v, σ_w, and q are optimized using a grid search on training datasets from WeAllWalk, aiming to minimize the weighted sum of the overcount and undercount rates for a two-stage detector. Given that turn undercounts can lead to more significant path reconstruction errors—since two consecutive incorrect turns of opposing angles may cancel each other out—a greater weight of 2.5 is assigned to the undercount rate in the optimization process.

3.4 Straight Walking (SW) Detector

The straight walking (SW) detector determines whether a user is walking ”regularly” on a straight path at each timestamp. The system is based on a Gated Recurrent Unit (GRU) [18], a variant of recurrent neural networks. This model, trained on the WeAllWalk dataset, employs a Leave-One-Person-Out cross-validation policy [39]. Taking azimuth and user acceleration magnitude (smoothed by a Gaussian filter with $\sigma = 15$) as inputs, this model outputs a probability of the user walking straight. Subsequently, the threshold yielding the highest f1-score on the training dataset is selected. Comparing the probability to this threshold, the detector's output is binary (0/1), where zero indicates straight walking.

Data within straight segments are labeled as SW, whereas data within other segment types (e.g., turn segments and feature sub-segments) are labeled as non-SW. Since participants in the WeAllWalk dataset typically remained stationary at the start and end of each trajectory, these sub-segments were manually labeled as non-SW. To accurately predict the label, the model must have access to adequate future data. Therefore, the GRU model makes its prediction with a 1.2-second delay, which has been empirically found to be effective. This method enables the model to learn the pattern of walking in a straight line, basing its decisions on a combination of both the azimuth pattern and the magnitude of the user's acceleration. The model, constructed using Keras and Python3, has an input window size of 150 samples and a hidden unit size of 32. Additionally, a dropout rate of 0.4 is set, and the model is trained over three epochs with a batch size of 2048. Figure 3.1 visualizes the SW detection results.

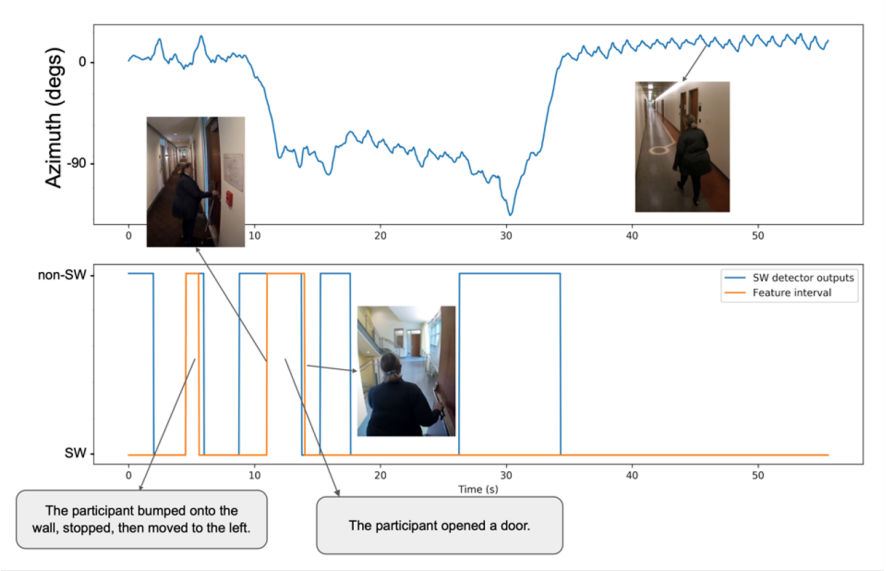


Figure 3.1: A visualization of the SW segment detection system. Top: Azimuth signal; Bottom: output of SW detector is shown in blue, and the feature intervals from WeAllWalk are shown in orange.

3.5 Optimizing Turn Detection

To enhance the navigation of visually impaired individuals, indoor walking trajectories can be visualized as a series of straight paths mixed with occasional turns. However, several scenarios may lead to erroneous turn detection for visually impaired users. For instance, a person may pause and keep changing directions to better perceive their environment, or they may execute a series of maneuvers, like turning left then right, to circumvent an obstacle. These movements can result in misleading turn signals, making the navigation instructions less effective in guiding the user back to their starting point. It's important to note that such false positives primarily occur during non-straight walking (non-SW) intervals, where changes in the user's facing direction are not of concern. To mitigate this, I analyze the orientation data from the Mixture Kalman Filter (MKF) model before and after non-SW intervals to obtain the actual

turn angle (see Figure 3.2). This method allows for the exclusion of incorrectly identified turns.

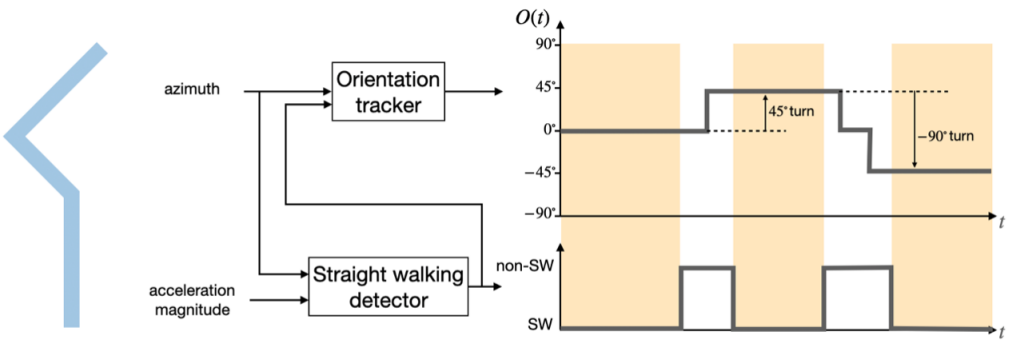
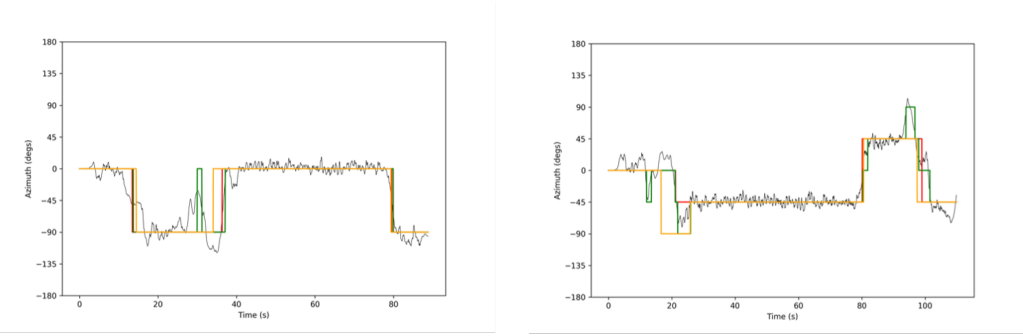


Figure 3.2: A visualization of the two-stage turn detector. The left blue line represents a user's trajectory. The user first walks straightly, then takes a 45° left turn and a -90° right turn. Middle diagram illustrates our two-stage orientation detection system structure. In right diagram, SW intervals are highlighted in yellow. Additionally, a turn is obtained by comparing the user-facing direction between two consecutive SW intervals.

It is assumed that user orientation remains constant during SW intervals; however, this may not always hold true due to the imperfections of our SW detector. For instance, orientation changes could occur at the start of an SW interval due to potential delays in our MKF model. To address this, the mode of the orientation distribution within each SW interval is computed to alleviate such discrepancies.

It has been observed that the MKF with a 90° orientation resolution demonstrates greater robustness compared to one that detects 45° turns. Within the WeAllWalk dataset, 13% of turns are ±45°. A 180° turn is identified by the MKF as either two consecutive ±90° turns or four consecutive ±45° turns. This detection is refined when comparing orientations between successive SW intervals. For example, two -45° turns would be detected as a single -90° turn by

our two-stage strategy in Figure 3.2. Examples of outputs from the two-stage turn detection system are illustrated in Figure 3.3. In Figure 3.3(a), notable variations in azimuth measurements around $t = 30$ seconds result in a series of detected turns. Since these orientation changes occur within a non-SW interval, our two-stage detection approach disregards these extra turns, yielding the correct turn angle. A comparable scenario is depicted in Figure 3.3(b) at approximately $t = 95$ seconds.



(a) (b)

Figure 3.3: An example of two-stage turn detection. Blackline: azimuth data. Greenline: MKF outputs, which is the orientation. Orange line: outputs of the two-stage turn detection system. Redline: user orientation ground truth provided by the WeAllWalk dataset. Outputs from the SW detector are not shown here for simplicity. Notice that the MKF orientation resolution is 90° (a) or 45° (b).

3.6 Conclusion

In this section, we address the challenge of detecting the direction of an indoor user in the absence of a map. To achieve this, I propose a two-stage model. The initial stage employs a Gated Recurrent Unit (GRU)-based detector to ascertain whether the user is proceeding in a straight line. The subsequent stage operates under the assumption that an indoor walker's turns

are typically in increments of 45 or 90 degrees. Here, a Mixture Kalman Filter (MKF) model is leveraged to correct for drift in the azimuth data. During periods identified as non-straight walking (non-SW), the probability of a turn is increased to better reflect real-world behavior. By integrating these two stages, this model creates a sequence of straight paths with turns. facilitating precise path reconstruction and navigation.

Chapter 4

Map-Assisted Localization

This section presents the problem of map-assisted localization. We will start by explaining the particle filter[68], which is the main algorithm used in our application for locating pedestrians inside buildings and outside in open environments. We will first discuss what a particle filter is, including how it works and its limitations. Then, we will go into how the particle filter is applied to localization tasks, particularly its role in combining data from different sensors and maps to improve the accuracy of determining a location. We will also look at how to handle situations where the particle filter detects multiple groups of data points.

4.1 Problem Statement

Though we are assuming that map data is available for this study, tracking people with visual impairments in both indoor and outdoor environments is still challenging. I want to clarify that we are not using cameras or other external devices. We no longer assume that the person is always in a corridor intersecting at 45° or 90°. This is because outdoor spaces can

be more complex and the idea of a fixed orientation doesn't apply. Also, using the map data provides a stronger constraint than just assuming a fixed orientation, so we can leave out the weaker assumption. The particle filter is the tool we use to maximize the utility of the map data. Let's start with a brief introduction to the particle filter.

4.2 Particle Filter

The particle filter is a type of Bayesian filter that proves highly effective in handling the complexities of real-world probability distributions, particularly in the context of incorporating map information. In the projects outlined within this thesis, floor plans and outdoor maps serve as constraints for the particle filter, enhancing its capacity to refine velocity errors and estimate user localization. This approach relies on a group of particles, with each one standing for a possible sample from the posterior distribution. The diversity of particles also helps in handling the uncertainties contained within sensor inputs, with each sensor presenting its own level of uncertainty that must be carefully managed.

The particle filter estimates the posterior distribution through a series of steps, using information from previous time steps and the current observation to estimate the current state. Specifically, it goes through predicting, updating, and resampling. We will also discuss the drawbacks of the particle filter later in this chapter.

4.2.1 Algorithm Steps

A standard particle filter operates through three main steps: the prediction step, the update step, and the resampling step. The details of these steps are as follows:

4.2.1.1 Prediction Step

This step predicts future states of the model using the current state and the necessary model input. No additional information is required. However, it is crucial to understand that predictions made in this step are not reliable on their own. We must adjust the weight of each particle by incorporating information from other sources to approximate the true distribution more closely.

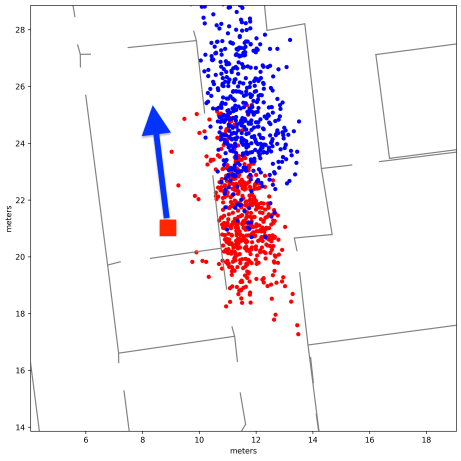
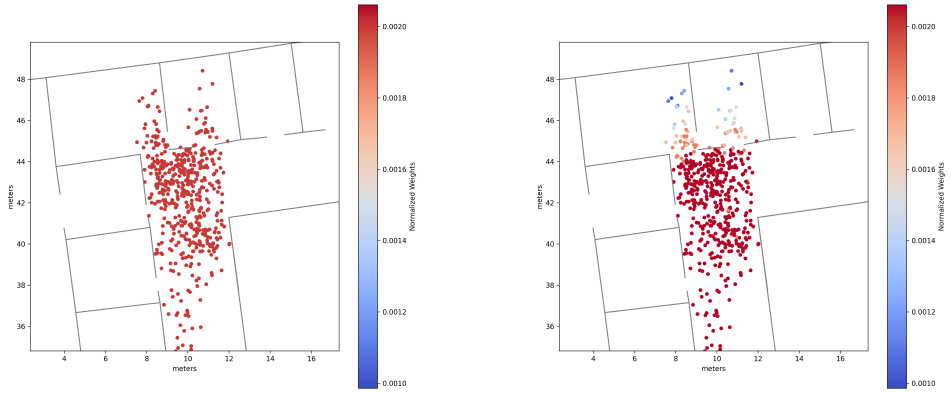


Figure 4.1: Visualization example of the prediction step of a particle filter. Red points: particle before update. Blue points: particle after update. The velocity originates from the red rectangle and its direction is indicated by the blue arrow.

4.2.1.2 Update Step

In the update step, we use data from various sources to update each particle's weight. We evaluate how well the state of each particle matches with the observed data. Particles that correspond closely with the observed data are given higher weights. It is also important to consider the reliability of each information source; sources with questionable reliability should have less impact on the particle weights compared to those that are considered highly reliable. If a source is entirely unreliable, it should be excluded. Conversely, if we are completely confident in the accuracy of a source, we might adjust the particles solely based on that information.

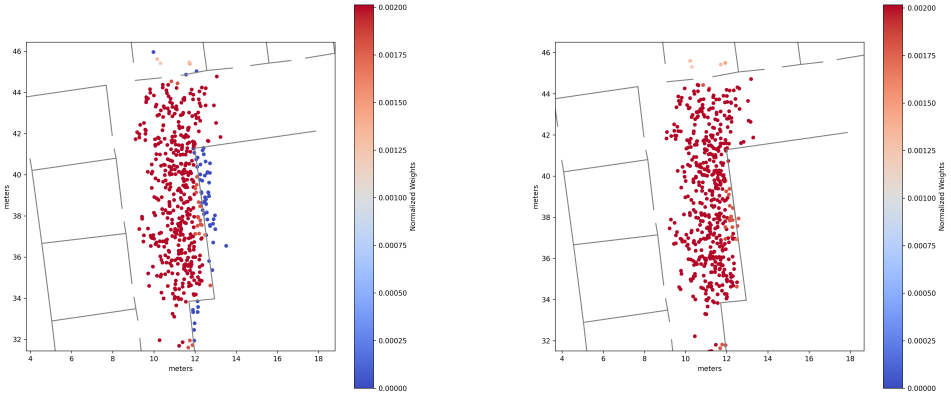


(a) (b)

Figure 4.2: Visualization of the update step of a particle filter. (a) Particles before update step. (b) Particles who stay in the room or goes deeper into the room get less weights, thus get different color.

4.2.1.3 Resampling Step

After updating the weights, we resample the particles based on their new weights, a process that can happen at different frequencies. This step focuses on the more promising particles—those with higher weights—by replacing those with lower weights. Although resampling does not necessarily occur after every update, it is an essential process for the particle filter to converge on an accurate distribution, a technique also known as importance sampling.



(a) (b)

Figure 4.3: Visualization of the resample step of a particle filter. (a) Particles before resample step. (b) Particles after resample step. Particles with low weights get removed.

4.2.2 Drawbacks

The particle filter, while a robust tool for managing non-linear systems, comes with several drawbacks that must be carefully managed for effective use. Let's discuss these limitations in detail.

4.2.2.1 Particle Degeneracy

Particle degeneracy occurs when almost all particles have very low weights except for a few with disproportionately high weights. This often results from infrequent resampling due to poor design choices. That is, particles with larger weights continue to dominate, while those with smaller weights diminish further. This would not be a concern if the high-weight particles accurately represented the real distribution, but this is seldom the case due to noisy input signals. Therefore, it's not wise to depend only on a few particles with high importance weights. A diverse set of particles with comparable weights is preferable, allowing the model to navigate uncertainties more effectively and estimate the true distribution more accurately.

4.2.2.2 Sample Impoverishment

Although resampling can mitigate particle degeneracy, it can introduce the issue of sample impoverishment. This occurs when particles have similar weights but are not diverse, limiting the model's ability to explore the true distribution. The lack of diversity arises when insufficient randomness is introduced during the prediction and resampling steps, resulting in particle states that are too alike.

4.2.2.3 Computational Cost

Particle filters can be computationally demanding, particularly when utilizing a large number of particles. This necessitates careful consideration of the particle filter implementation, balancing performance against computational costs to optimize the model's efficacy.

The optimization of particle filter lies on nearly every aspect of the algorithm. Here

we are going to highlight the reason why the particle filter can be time consuming. The solution, or speed up methods will be discussed in more details in later sections. Now let's begin with how we design the usage of this algorithm. More specifically, we assume the user starts at a position instead of spreading particles everywhere, this significantly decrease the computational cost. For particle filter, it's going to be a problem if it takes too much time to perform the update steps of each particle because we need to multiply the consumed time with the particle number at each iteration. The more particles we have, the more time it consumes. For instance, it's questionable when one want to figure out whether each particle trajectory bumps into a wall. Additionally, when dealing with GPS, given the GPS model, the code used to calculate the necessary PDF for the weights update must be optimized for efficiency, which can be achieved by using SIMD to implement the PDF calculation function. Additionally, one may not want to do the resampling process too frequently because it's also time-consuming. In fact, if there are too many prediction and update steps in each second, one may want to use metric such as ESS (Efficient Sample Size) to decide when to perform a resampling. Also, the particle filter may have multiple cluster problem, and the mean-shift algorithm is needed to handle that. The speed up of the mean-shift algorithm is another topic needs to be taken care of.

4.3 Particle Filter-Based Localization

In the preceding section, we examined the advantages and disadvantages of the particle filter. We will now shift our attention to the practical application of particle filters in the real world, particularly for localization tasks. This section will present a more generalized dis-

cussion on particle filter-based localization, applicable to both indoor and outdoor scenarios, comparing two distinct methodologies and assessing their strengths and weaknesses.

4.3.1 Implementing Particle Filters for Localization

A common approach in particle filter application is for each particle to represent a 2D position within a map's floor plan. Initially, these particles are spread across the entire map area. As the system receives observations from various sensors, it updates the weights of the particles. After several iterations and the necessary resampling, one or more clusters emerge, pinpointing the user's current location. However, the computational demands of this method are significant, as distributing and continually updating a large number of particles can be resource-intensive, which is not suitable for smartphone applications. Reducing the number of particles is not a viable solution either, as an inadequate density of particles may fail to capture the map's characteristics. Additionally, the convergence time and the likelihood of identifying multiple clusters when the information is incomplete or inaccurate make this approach less suitable for navigation tasks that require immediate feedback.

Therefore, the localization project developed in this thesis employs an alternative use of particle filtering. By assuming a known initial point and walking direction, the model updates the position of all particles by integrating data from the smartphone's inertial sensors, adjusting their weights according to other available sensors and map data. This approach ensures a sufficient distribution of particles around the user, facilitating a thorough examination of nearby walls, rooms, or landmarks without exhausting the computational resources available on a smartphone platform.

Though the first method has the benefit of not necessitating a known starting point, acquiring such a starting point and initial walking direction is typically not problematic for visually impaired users. Finding a recognizable location, such as an elevator, building entrance, or landmark, and following along a wall can provide the essential initial data. Additionally, while the first method can autonomously estimate initial starting points, in the case of an exceedingly large map, a subarea must still be provided to manage the overall number of particles effectively, thereby requiring a rough starting point.

4.4 Multiple Clustering Problem

For both indoor and outdoor scenarios, an important issue that must be managed when using particle filtering is the multi-cluster problem. Typically, particles will converge in proximity to one another. However, there are instances, especially when some particles do not accurately reflect the real environment, where this is not the case. For instance, some particles may advance significantly faster than others, or a large cluster may be divided by a wall after the user takes a turn. In such situations, two or multiple clusters may form, each representing a potential location in the particle filter's estimation. Figure 4.4 illustrates two clusters of particles at a specific moment in time.

To address this issue, the initial step is to determine the number of clusters present. There are several off-the-shelf cluster detection algorithms available, such as k-means[1]. However, k-means requires pre-setting the number of clusters, which is not feasible in an online context. Although there are techniques to adjust the k value dynamically [64], they add unnec-

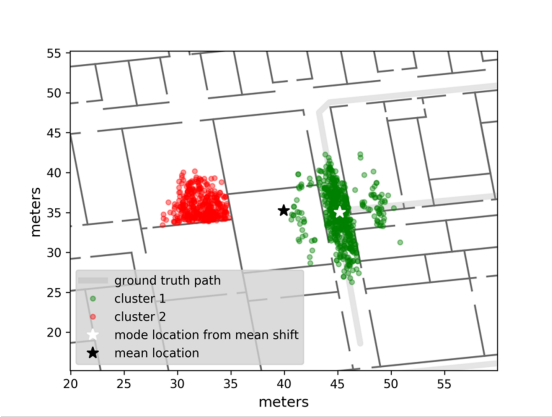


Figure 4.4: This figure visualizes a set of particles at a certain time. The two main clusters are depicted in red and green. The posterior mean, shown in black, is incorrect due to the bi-modal posterior distribution. The highest distribution mode identified by the mean shift algorithm is marked with a white star.

essary complexity to the system. Therefore, in my projects, the mean-shift algorithm has been adopted.

4.4.1 Mean-Shift Algorithm

The mean-shift algorithm [16] is a clustering technique that processes a set of particles and automatically assigns each particle to a cluster. Unlike k-means, the number of clusters is not predetermined but is instead determined by the mean-shift algorithm as an output. However, this algorithm has its own trade-offs. To utilize mean-shift, the user must specify a critical parameter referred to as the bandwidth. This parameter defines the neighborhood size within which a particle searches for its neighbors. Once the bandwidth is established, the algorithm proceeds as follows. Firstly, each particle's neighbors within a certain bandwidth are identified. We calculate the mean position of these neighbors. This mean then becomes the new reference point for the next iteration, where we identify neighbors around this new mean and recalculate

it. This process repeats until the mean stabilizes or reaches a maximum iteration count. In this manner, a mean position is determined for each particle. The subsequent step involves removing mean points that are close to each other, resulting in a set of mean points that are well-separated. Each of these mean points can be considered a cluster center. Following this, we assign each particle to its nearest mean point, thereby assigning each particle to a cluster. It's important to note that the bandwidth value can be adjusted based on the specific environment. For example, a 5-meter bandwidth might be suitable for indoor scenarios, while a 7-meter bandwidth could be more appropriate for outdoor environments. The detailed mean-shift algorithm are shown in Algorithm 2. One very important thing is to speed up the meanshift algorithm since we need to run this in realtime on an iPhone. For the original mean-shift algorithm mentioned above, it used every position as a candidate to calculate the cluster mean center. However, it's not possible to do that frequently in an iPhone. Thus, as we can see in Algorithm 2, we deploy the normally used method: we do the sampling with a grid.

4.4.2 Calculating the Final Position Output

After acquiring cluster information, the next task is to select which cluster should be used to determine the output position. The previously mentioned weighted mean method is effective when only one cluster is detected, but it fails with multiple clusters, resulting in a position that may fall in an inaccessible area. I introduce two algorithms to derive the final output position from multiple clusters: the local-maximum and global-maximum algorithms.

The local maximum algorithm, while intuitive, is quite popular. It involves selecting the cluster with the most particles and using its weighted mean as the output position. Alterna-

Algorithm 2 Modified Mean-Shift Clustering

Define Parameters:

bandwidth: The radius of the window used for mean shift. Adjust according to different scenarios.

MaxIter: Maximum number of iterations to stop the algorithm if it does not converge.

Select Bin Points:

Select a subset of points (*bin_points*) from all points using a binning method. The bin size is set equal to the *bandwidth* value.

Mean Position Calculation:

for each candidate point in *bin_points* **do**

    Initialize iteration count *iteration* to 0.

    repeat

        Calculate the mean position of all points within *bandwidth* of the candidate point.

        Update the candidate point to this mean position.

        Increment *iteration* by 1.

    until distance between new and previous candidate positions is less than $1e-3 \times$ *bandwidth* or *iteration* exceeds *MaxIter*

end for

Remove Near-Duplicate Centers:

Remove cluster centers that are within a small threshold distance of each other.

Assign Points to Clusters:

Assign each point in the original dataset to the nearest cluster center determined by the mean shift process.

tively, the cluster with the highest cumulative weight may be chosen. The primary advantage of this method is its simplicity and ease of implementation. However, a potential drawback is that the largest cluster may not always correspond to the actual user location, regardless of its size.

Now, let's delve into the global maximum algorithm. By continuously tracking each particle's cluster transitions, we can refine the entire particle trajectory based on the current largest cluster. In practice, given the current local maximum, one can trace the origins of all particles within that cluster. This process yields a more accurate historical trajectory, which can be valuable for applications such as trajectory analysis. Note that for real-time applications concerned only with the user's current position, the global maximum algorithm provides the same output as the local maximum since it returns the largest cluster at that instant. However, the global maximum algorithm can enhance previous location estimates, unlike the local maximum method.

For indoor localization, the weighted mean of all clusters is currently used since no significant extra clusters have been observed during user studies. For outdoor localization, the local maximum algorithm is employed to yield better localization results when multiple clusters are detected.

4.5 Conclusion

This section has concentrated on the algorithm for map-assisted localization. We started with the fundamental concept of the particle filter, which involves prediction, update, and resampling steps. We then explored its drawbacks, such as particle degeneracy, sample

impoverishment, and computational cost. Next, we discussed two approaches for the particle filter implementation. Subsequently, we introduced an off-the-shelf clustering algorithm named mean-shift and described two methods to leverage the clustering results, namely the local maximum and global maximum algorithms.

Chapter 5

Particle Filtering for Indoor Localization

In this section, we explore into the application of particle filtering within indoor environments. We will discuss the smartphone sensors that are available for indoor use, explore the common states used in indoor localization scenarios, and provide detailed descriptions of the particle filter steps, including special considerations for handling extreme cases.

5.1 Available Data

In this section, we will focus on the data sources used in indoor environments, particularly emphasizing the role of map data and the iPhone's Inertial Measurement Unit (IMU) sensor.

5.1.1 Map Information

Map information plays a crucial role once it is available, offering strong constraints to mitigate potential drift errors from IMU sensors. The maps used in this research are in the

geojson format, supplied by the SIM web application[1] [69]. An online service, which will be introduced later in this thesis, converts the map into a specific format suitable for the app. Notice that direct usage of geojson files is avoided due to the need for specialized handling; particularly, during the particle filter update process, it is necessary to ascertain whether a particle's trajectory intersects with any walls. A straightforward approach would be to compare each trajectory against all walls, but this is computationally prohibitive. Assuming the number of particles is N and the number of walls is M, the computational complexity for each update step would be $O(NM)$, which is typically $O(N^2)$ when N and M are large and of similar value.

To resolve this, a trade-off between time complexity and space is made. Initially, the map is rendered onto a 2D matrix using the Bresenham algorithm [11], where wall segments are marked with a value of 1. Subsequently, each particle's trajectory is also drawn onto this matrix, checking for any overlap with wall segments. Since the length of trajectories is significantly less than M, the overall time complexity is reduced to $O(N)$, making particle filter execution feasible on a smartphone. Other matrices are similarly created to efficiently represent rooms and corridors.

5.1.2 IMU Sensor

The range of sensors available on a smartphone for indoor applications is limited. The onboard IMU sensor, which provides acceleration and rotation rate data, is utilized in this study. More specifically, we have two models in use: the pedestrian dead-reckoning model and the RoNIN model, which were introduced in earlier sections.

[1] `https://sim.soe.ucsc.edu`

The iPhone provides multiple types of IMU sensor data, including the raw acceleration and gyroscope data, which can be obtained from CMMotionManager [2], namely the accelerometerData and gyroData fields. These data are used as input to the RoNIN model, and are collected in 200Hz. Another choice is the userAcceleration and rotationRate from the CMdeviceMotion [3]. Additionally, the attitude property of CMdeviceMotion provides rotation matrix R, which can be used to extract the azimuth angle:

$$Azimuth = atan2(R.m12, R.m11) \tag{5.1}$$

The azimuth, userAcceleration are used as inputs to the straight walking detector, and the userAcceleration with rotationRate are used as inputs to the step counter model. These data are collected in 25Hz.

5.2 States Definition

The state of an individual particle is defined as follows, where t is the current timestamp and i is the index of the particle. The variables x_t^i and y_t^i represent the particle's current position, d_t^i denotes the drift, s_t^i is the stride length, and w_t^i is the particle's weight.

$$state_t^i = [x_t^i, y_t^i, d_t^i, s_t^i, w_t^i] \tag{5.2}$$

It should be noted that stride length is necessary if pedestrian dead-reckoning is employed; however, it is omitted if the RoNIN model is used. Let us examine these states in greater detail:

[2] https://developer.apple.com/documentation/coremotion/cmmotionmanager
[3] https://developer.apple.com/documentation/coremotion/cmdevicemotion

1. **Position**: Each particle's state includes its current position on the 2D map, representing a potential location of the user.

2. **Drift**: To account for azimuth data drift, each particle maintains a drift value.

3. **Stride Length**: For pedestrian dead-reckoning, an initial stride length is obtained through calibration for each user. While a constant stride length could be used, its reliability is questionable due to variations in walking patterns between calibration and actual use. Therefore, a better approach is to allow the particle filter to estimate stride length dynamically. Each particle starts with a stride length perturbed by Gaussian noise from the calibrated value, allowing the particle filter to converge on a more accurate estimate as the user moves, particularly after turns. A similar approach to estimating stride length using a particle filter can be found in [5].

4. **Weight**: The weight of each particle is a scalar indicating its relative importance or our confidence in it. This weight is updated during the update and resampling steps, which will be discussed later in this section.

5.3 Implementation Details

Now, we'll explore the details of implementing the particle filter in indoor environments. We'll first discuss the three main steps: the prediction step, update step, and resampling step. After that, we'll examine the extreme scenarios that occur in indoor settings, using examples to illustrate these cases.

5.3.1 Prediction Step Details

In the prediction step, velocity data from inertial sensors serve as the input. This velocity is derived from a reconstructed path, which has been aligned with the app's coordinate system during an initial calibration phase. The reconstructed path may originate from either the RoNIN model or another pedestrian dead-reckoning model. Velocity is computed by calculating the distance between two consecutive points on the reconstructed path and dividing by the time elapsed.

With the velocity obtained, the next step is to update the position of each particle. We first introduce random walk Gaussian noise to the drift state. Then, we introduce Gaussian noise to the magnitude and direction of the input velocity before updating the particle's position, accounting for the drift in the angle calculation. This ensures diversity in the drift values across particles and compensates for the uncertainty in velocity input. Here are the formulas:

$$d_t^i = \mathcal{N}(d_{t-1}^i, \sigma_d^2) \tag{5.3}$$

$$x_t^i = x_{t-1}^i + (v_m \cdot \Delta t + \mathcal{N}(0, \sigma_m^2)) \cdot \cos(\mathcal{N}(v_a, \sigma_a^2) + d_t^i) \tag{5.4}$$

$$y_t^i = y_{t-1}^i + (v_m \cdot \Delta t + \mathcal{N}(0, \sigma_m^2)) \cdot \sin(\mathcal{N}(v_a, \sigma_a^2) + d_t^i) \tag{5.5}$$

The elapsed time since the last iteration is represented by Δt, with v_m and v_a denoting the velocity magnitude and angle, respectively. The variances for the Gaussian noise applied to the drift, magnitude, and angle are given by σ_d^2, σ_m^2, and σ_a^2, respectively.

5.3.2 Update Step Details

The update step adjusts the weights of the particles based on the map information. Several assumptions about the map are made to optimize the use of this information:

1. Doors on the map are assumed to be open, maintaining the possibility that a user could mistakenly enter a room. This assumption aligns the model more closely with real-world scenarios and was validated during the user study. Since room entry is not encouraged, as the focus is on guiding the user through corridors, particles that enter a room have their weights reduced by a factor of 0.9 at each time step.

2. Particles are not permitted to pass through walls, reflecting the impossibility of a person doing so. Particles with trajectories intersecting walls are identified using the 2D map matrix and are assigned a weight of zero, which results in their removal during the immediate subsequent resampling step.

3. Particles that move outside the building are also assigned a weight of zero, reflecting the focus on indoor localization. Additionally, for users on floors above the ground level, it is impossible for them to exit the building boundary.

5.3.3 Resampling Step Details

Resampling is a critical step that aligns the proposal distribution closer to the target distribution. resampling occurs under the following conditions:

1. If any particles collide with a wall, they are immediately replaced with a 'healthy' particle that has survived without collision.

2. resampling is performed immediately when particles enter a room to discourage this movement and to make instant use of the room information.

Given these considerations, I have decided to execute the resampling process at each step for indoor navigation. This is done using a low-variance sampling algorithm, which is computationally efficient and avoids overburdening smartphone CPUs.

Prior to and following resampling, weights are normalized to ensure the sum of all particle weights equals one. This normalization is straightforward: each particle's weight is divided by the total sum of weights. For N particles, the normalized weight of particle i, w_t^{i*}, is calculated as follows:

$$w_t^{i*} = \frac{w_t^i}{\sum_{j=1}^{N} w_t^j} \tag{5.6}$$

Once resampling is complete, the particle filter output is ready. Typically, the weighted mean of all particle positions is used as the output location. This approach is logical since the weight of each particle signifies its relative importance, with particles of higher weight contributing more significantly to the final result.

5.3.4 Extreme Case Handling for Indoor Scenario

Although the particle filter model is generally efficient, extreme cases in practical implementation necessitate additional considerations to ensure real-world applicability. These cases often stem from the inherent limitations of particle filtering, which have been previously discussed. Here, we focus on how these limitations appear in indoor localization scenarios and propose potential solutions.

5.3.4.1 Environmental Modeling Errors

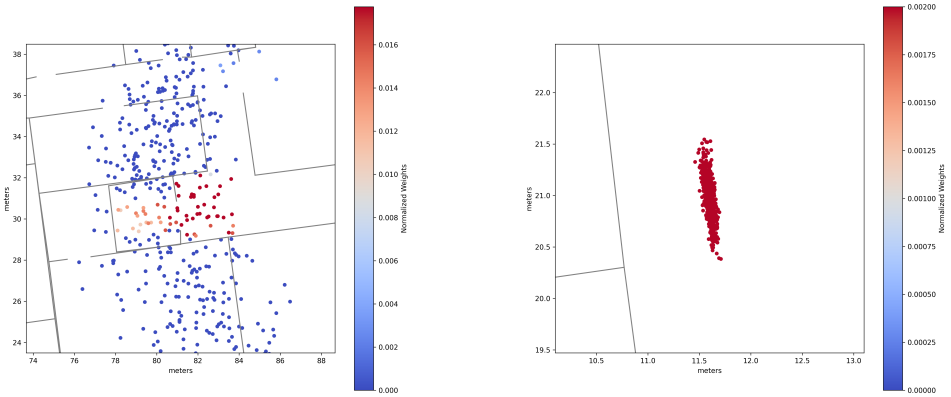
Occur when the particle filter fails to accurately represent the environmental distribution. An example is when all particles erroneously exit the building, resulting in zero-weight particles and an empty list, which halts the model. A fail-safe mechanism is necessary to counter this. Another issue is particles getting trapped in a corner, continuously colliding with a wall and being resampled, which hinders further progress. Introducing positional noise during resampling can provide trapped particles with an opportunity to escape and return to the correct path.

5.3.4.2 Particle Degeneracy

This situation can happen when most particles enter a room, leaving only a few in the corridor, those in the corridor may dominate weight distribution (see Figure 5.1(a)). High-frequency resampling can prevent this imbalance, promoting a more even distribution of particles throughout the corridor to explore the map effectively.

5.3.4.3 Sample Impoverishment

This situation can happen when no noise is added during the prediction step. One example is that, all particles start with same initial value, with same velocity input will certainly went to the same position, and their weights will be updated in the same way. In this case, even though each particle gets equal weights, the power of particle filter doesn't really release, since its results is similar to just using a single particle. The correct way to deal with this is to add enough noise during the prediction step. Figure 5.1(b) illustrates an example of sample



(a) (b)

Figure 5.1: Visualization example of two different particle filter problem. (a) Visualization example of the particle degeneracy problem. Only a small number of particles that lie in the corridor has large weights, while all other particles spread into different room and has low weights. In this scenario, it becomes challenging to determine the user's true position, resulting in the wastage of a majority of particles. (b) Visualization example of the Sample Impoverishment. The particles are closely clustered together and positioned in the center of a corridor, indicating an inefficient method for detecting map information.

impoverishment caused by insufficient noise added during the prediction step.

5.3.4.4 Long Corridor Problem

Long corridors or open spaces present unique challenges for particle filtering due to the limited map information available. While particle diversity is desired, excessive spreading without constraints can be problematic. Implementing a limit on how far particles can move away from the cluster center can prevent them from spreading too widely, ensuring they remain within a relevant range for resampling.

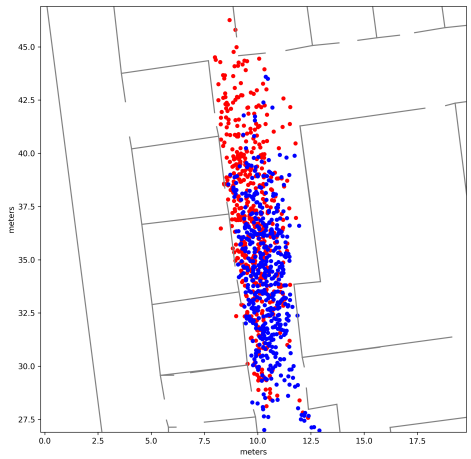


Figure 5.2: Visualization of the long corridor problem. Red points: Particles spread over the long corridor without any constraints. Blue points: With constraints, particles won't be too far away from its cluster center.

5.4 Conclusion

This section has centered on the particle filter approach to indoor localization. We began by examining the data sources available, including maps and IMU sensor data. We discussed the state components for two different PDR methods, which include user position, drift, weight, and stride length. Detailed implementation steps for prediction, update, and resampling were addressed, highlighting the use of IMU sensor data to move particles during prediction and the use of map data to adjust particle weights during updates. The resampling process is executed at each iteration with normalization of particle weights. Finally, we explored multiple extreme cases—environmental modeling errors, particle degeneracy, sample impoverishment, and the long corridor issue—along with their respective solutions.

Chapter 6

Particle Filtering for Outdoor Localization

In this section, we will discuss the application of particle filtering in outdoor localization scenarios. While the core structure of the particle filter remains similar to the indoor case, additional sources of information are available, and unique challenges must be addressed. We begin by examining the sensors available for outdoor localization, including the advantages and disadvantages of GPS signals. We then detail the specific steps of the particle filter tailored for outdoor use. Finally, we explore the extreme cases that require careful handling in outdoor localization tasks.

6.1 Available Data

Localizing users outdoors presents more complex challenges compared to indoor scenarios. Along with map data, sensor inputs from IMU, GPS, and altimeters are available. This subsection discusses the characteristics of these sensors, along with their benefits and limitations.

6.1.1 Map Information

The use of map information in outdoor environments follows a similar approach to that of indoor localization, such as employing the Bresenham algorithm for efficiency. However, outdoor maps typically offer less detail and fewer constraints compared to indoor maps. The lack of structural constraints like walls and landmarks means that drift errors from IMU sensors cannot be accurately estimated using map data alone. Consequently, additional sensor inputs, particularly GPS signals, are essential for achieving satisfactory localization results.

6.1.2 GPS Signal

As we discussed in subsection 2.1.6, GPS provides the user's absolute location on earth estimated from distances to multiple satellites. This technology is embedded in many modern devices, including smartphones, and is especially useful outdoors where satellite signals are clear. However, GPS availability diminishes indoors or in locations like underground tunnels where signals are blocked or highly reflected. Our system utilizes the iPhone's Core Location API [1] to obtain positions and their associated uncertainty radius. Actually, the positions provided by Core Location are a combination of data from various sources, including GPS, cellular, and Wi-Fi. Additionally, this API does not provide the raw GPS satellite signal. For the sake of clarity and readability, we will refer to this localization data as "GPS" throughout this thesis. The GPS signal is crucial for the following usages:

1. Initial rough positioning for system initialization, especially when starting points and

[1] https://developer.apple.com/documentation/corelocation/getting_the_current_location_of_a_device

walking directions are not easily identifiable by blind users outdoors.

2. Providing additional information in open spaces where map data is insufficient for particle filter and IMU tracking.

3. Offering absolute positioning to complement the relative positioning from the particle filter and IMU system, aiding in system correction when necessary.

However, GPS signals can exhibit large uncertainty radius, which may be insufficient to navigate visually impaired users (Figure 6.1(a)). Furthermore, GPS reliability can vary, even with a small reported uncertainty (Figure 6.1(b)). These characteristics necessitate cautious treatment of GPS data.

(a) (b)

Figure 6.1: GPS Outage Example: The map serves as the background. The blue star represents the actual user position, the blue dot indicates the GPS position, and the blue circle represents GPS uncertainty from the smartphone API. (a) In this scenario, the GPS uncertainty is too large, making it unsuitable for navigation. (b) Despite a small GPS uncertainty, the provided position is far from the actual user position, making it unreliable for navigation.

6.1.3 IMU Sensor

As in the indoor scenario, IMU sensor data is crucial for the particle filter's prediction step. For outdoor localization, we focus on the RoNIN model, as previously detailed in subsection 2.2.3.

6.1.4 Altimeter

Altimeter data from the iPhone is essential for applications involving navigation through underground tunnels. It helps determine when to switch from overground to underground maps based on altitude changes. However, pinpointing the exact moment for this switch is challenging due to the variability and bias inherent in altitude measurements. Further discussion on the use of altimeter data will follow in section 9.4.

6.2 States Definition

The state definition for outdoor localization closely mirrors that of the indoor particle filter. The following state vector includes the position, drift, and weight of each particle:

$$state_t^i = [x_t^i, y_t^i, d_t^i, w_t^i] \tag{6.1}$$

For more detailed information, please refer to Chapter 5, which covers indoor localization.

6.3 Implementation Details

The implementation details for the outdoor particle filter share similarities with those in the indoor case, yet there are notable distinctions. For example, in the update step, the

availability of the GPS signal is a key factor, and its utilization is explored in this section. We also introduce an algorithm to adapt to uncertainty values. Furthermore, this section delves into additional extreme cases that are encountered in outdoor scenarios.

6.3.1 Prediction Step Details

The prediction step in the outdoor particle filter closely resembles the indoor approach. However, the parameters, such as the noise applied to the velocity input, require careful adjustment to reflect the different environmental factors present outdoors.

6.3.2 Update Step Details

While we continue to remove particles that collide with walls, the update step in the outdoor particle filter significantly differs from the indoor process, primarily due to the inclusion of GPS data and the absence of room boundaries in outdoor settings, necessitating careful integration.

6.3.2.1 Integrating GPS Signal

For implementation, the GPS data is modeled as a bi-variate Gaussian distribution. Updating the weight of each particle involves calculating its probability density function (PDF) from the GPS distribution, depending on the particle's distance from the GPS position and the GPS uncertainty. This process evaluates the alignment between the particle filter states and the GPS sensor readings, adjusting particle weights accordingly. The position of the i^{th} particle is

denoted by:

$$X^i = \begin{bmatrix} p_x^i \\ p_y^i \end{bmatrix} \tag{6.2}$$

The GPS position and uncertainty are represented by:

$$\mu = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \tag{6.3}$$

$$\Sigma_g = \begin{bmatrix} \sigma_g^2 & 0 \\ 0 & \sigma_g^2 \end{bmatrix} \tag{6.4}$$

The PDF for updating the weight of particle i is given by:

$$p_i(x) = \frac{1}{\sqrt{2\pi|\Sigma_g|}} \exp\left(-\frac{1}{2}(X^i-\mu)^T\Sigma_g^{-1}(X^i-\mu)\right) \tag{6.5}$$

It is important to note that the uncertainty value should not be taken directly from the smartphone platform's reported GPS uncertainty, as it may not be reliable. Instead, the uncertainty in our bi-variate Gaussian model should be set based on our confidence in the GPS sensor. A mechanism to determine a robust GPS uncertainty value, resistant to GPS errors, will be introduced as follows.

6.3.2.2 GPS Uncertainty Value Adaptation

According to our practical observation, a known issue with GPS is the occasional erroneous jump to a distant location, followed by a return to the correct position after several timestamps. This behavior necessitates a distinction between two cases for handling GPS data:

First, when the GPS position is significantly incorrect, particles tend to be updated with similar weights, as they are all distant from the GPS position. In such instances, no additional handling is required since the particles are largely unaffected by the distant, inaccurate GPS signal.

Second, when the GPS position is only slightly off before returning to an accurate state, particles coincidentally near the erroneous GPS location receive higher weights, skewing the trajectory. This more subtle GPS error is the focus of this section and requires a refined approach to ensure robustness.

The goal is to limit the influence of GPS on particle weights when GPS data is suspect. We employ an adaptive algorithm to dynamically adjust the GPS uncertainty based on observed behavior. This method operates under two assumptions:

1. GPS data may occasionally be incorrect but will return to an accurate value eventually.

2. The erroneous GPS position will not be extremely distant from all particles.

The proposed method limits GPS influence by defining a dynamic uncertainty value at each timestamp. Rather than directly setting a distance threshold, we establish a particle number threshold N_m, representing the number of particles beyond a certain distance D from the GPS location. By controlling N_m, we indirectly set the influence range of the GPS signal. Let N_0 be the number of particles observed beyond the distance D. If N_0 exceeds N_m, it suggests GPS inaccuracy, and D is increased to maintain N_m particles within the influence range. In cases where N_0 is less than N_m, it implies that the GPS accuracy is within an acceptable range, and no further adjustments are needed.

Algorithm 3 Estimate GPS Adaptive Uncertainty Value

1: Sort all particles by their distance from the GPS location in descending order.

2: Calculate N_0, the count of particles at a distance greater than σ, the last known GPS uncertainty radius set in the previous timestamp.

3: Identify the $\min(N_m, N_0)$-th particle in the sorted array. Let D be this particle's distance to the GPS location.

4: Update σ to equal D.

5: Apply weights to all particles using a Gaussian distribution with σ centered on the GPS location.

6: Adjust N_m using an exponential filter: $N_m \leftarrow \alpha \cdot N_0 + (1-\alpha) \cdot N_m$.

As we can see in the Algorithm 3, the last step employs an exponential filter to update N_m. α, in this context, determines how quickly the parameter N_m responds to changes in the GPS signal behavior. In practice, it's advisable to choose a small value for α. This adjustment is key because N_m should not remain static; it must respond to the ongoing GPS signal behavior at a moderate rate. The exponential filter functions as follows: After a GPS jump, a large N_0 value will occur since current particles are aligned with the pre-jump GPS signal. There are two possible outcomes:

1. If the GPS signal is incorrect but corrects itself quickly, N_m remains relatively stable due to the small alpha value, limiting the impact of the erroneous N_0. Consequently, the GPS uncertainty D will be larger, reducing the influence on particles. Once the GPS returns to accuracy, N_0 will decrease, maintaining a stable N_m.

2. If the GPS signal is accurate after a jump, it's necessary for particles to converge towards the new GPS location. N_m will update continuously based on N_0 until it matches N_0, indicating reduced GPS uncertainty and prompting particles to move towards the GPS position.

6.3.3 Resampling Step Details

The resampling process in the outdoor particle filter shares similarities with its indoor counterpart but also exhibits distinct differences. Notably, resampling does not occur at every iteration due to the presence of open spaces where particles are less likely to encounter physical barriers after a prediction step. Instead, resampling is conditional, based on the diversity of particle weights as measured by the effective sample size (ESS) [53], defined as:

$$E_t = \frac{1}{\sum_{i=1}^{N} (w_t^i)^2} \tag{6.6}$$

The ESS equals the number of particles N when weights are uniformly distributed; its value diminishes as weight diversity decrease. Resampling is performed when ESS falls below half the number of particles.

6.3.4 Extreme Case Handling for Outdoor Scenario

Handling extreme cases in the outdoor environment follows a similar framework to that of the indoor scenario, but with additional considerations for managing unreliable GPS signals.

6.3.4.1 Handling Extreme GPS Signal

While the adaptive algorithm to adjust GPS uncertainty is effective, GPS can still be consistently unreliable in certain areas, like the entrance of underground tunnels where signals are partially blocked. A practical solution is to ignore GPS data in known areas of consistent unreliability.

6.3.4.2 Open Space Problem

Localization in open spaces is particularly challenging due to the lack of map constraints. In such situations, the particle filter relies solely on IMU and GPS data. If GPS signals are precise, particle spreading is limited, but with large GPS uncertainty, the particles may spread everywhere until more constraints occur. This issue may be mitigated through user interface design, such as prompting users to locate nearby landmarks to recalibrate particle positions. However, it is important to acknowledge that providing accurate localization based solely on phone-based inertial sensors remains an open problem that necessitates further research.

6.4 Conclusion

This section has explored into the utilization of a particle filter model for outdoor localization. We began by discussing critical data sources for outdoor environments, particularly GPS, which facilitates rough initialization, aids in navigating open spaces, and provides absolute positioning to complement the relative positions from the particle filter. We reviewed the model's states and three key steps, with special attention given to the integration and adap-

tive estimation of GPS uncertainty to address the challenge of GPS signal jumps. The concept of effective sample size was introduced as a metric for assessing weight diversity and determining the need for resampling. Lastly, we examined two extreme cases in outdoor scenarios: unreliable GPS signals and the challenge of open space navigation.

Chapter 7

Experiments on the WeAllWalk Dataset

In this chapter, we present the experimental results of our methods applied to the WeAllWalk dataset. Initially, we explore various test and training modalities. Subsequently, we evaluate the performance of the two-stage turn detector on the WeAllWalk dataset. Furthermore, we introduce evaluation metrics for the reconstructed path using our methods. We assess the results in two scenarios: the map-less case and the map-assisted cases. The outcomes are visualized on map images for a thorough analysis.

7.1 Training/Testing Modalities

Given the varying walking characteristics between long cane users and dog guide users [35], we approached the training and testing modalities separately for each group. The following schemes were considered:

1. **Train on Sighted (TS).** In this scenario, the model is exclusively trained using data from

sighted individuals. Subsequently, the model's performance is evaluated using data collected from either long cane users (TS: LC) or dog guide users (TS: DG).

2. **Train in the Same Community (TC).** Three distinct communities are taken into account: long cane users (TC: LC), dog guide users (TC: DG), and sighted users (TC: S). Within each community, a Leave-One-Person-Out cross-validation policy [39] is employed. This involves training and evaluating the model using data from different individuals within the same community.

3. **Train on All (TA).** The training dataset comprises all available data from WeAllWalk, and a Leave-One-Person-Out cross-validation policy is applied (TA:LC, TA:DG). For instance, if a dog guide user is part of the test set, the model is trained using data from all sighted participants, all long cane users, and all other dog guide users.

For the subsequent modality tests, we calculate the average of the relevant measurements across both iPhones carried by the walkers, all paths, and all participants in the test set to obtain evaluation results.

7.2 Turn Detection

To evaluate the performance of our proposed two-stage turn detector, we employ the Longest Common Subsequence (LCS) algorithm. Specifically, we compare two sequences of turns and identify the longest ordered matching subsequences. Subsequently, we compute the overcounts (OC) and undercounts (UC). These counts are then normalized by dividing them by the ground-truth number of turns to obtain the overcount rate and undercount rate. It's worth

noting that the 90° turn detector is incapable of detecting 45° turns. Therefore, any potential overcounts or undercounts attributed to this limitation are excluded from the analysis. The results are presented in Table 7.1.

Error Type	**45° TD-Error**		**90° TD-Error**	
	UC rate %	**OC rate %**	**UC rate %**	**OC rate %**
TS:LC	0.64	6.85	0	1.87
TS:DG	1.14	4.49	0.81	0.81
TC:S	0	0	0	0
TC:LC	1.64	3.98	0.54	0.26
TC:DG	**1.11**	**4.30**	**0**	**0.79**
TA:LC	**0.37**	**3.51**	**0**	**0.79**
TA:DG	1.15	5.39	0	0.83

Table 7.1: Turn detection (TD) error is reported for both our 45° turn detector and 90° turn detector. Within each community of blind walkers (LC, DG), the UC rate and OC rate pair with the smallest sum is highlighted in boldface.

As depicted in Table 7.1, our two-stage turn detector yields perfect results for sighted walkers. Notably, the 90° turn detector exhibits lower error rates compared to the 45° turn detector. Furthermore, among long cane users, the model trained using all available data (TA: LC) outperforms the one trained solely with data from sighted individuals (TS: LC). For dog guide users, the model trained with data from dog guide users (TC: DG) demonstrates the most favorable results.

7.3 Path Reconstruction

In this section, we introduce three evaluation metrics that will be used to assess the paths reconstructed by various algorithms, both in the map-less and map-assisted cases.

7.3.1 Evaluation Metrics

The WeAllWalk dataset provides timestamps t_j^i corresponding to when walkers pass each waypoint. To approximate the ground-truth trajectories, we assume that each user was positioned in the middle of the corridor when passing each waypoint. We measure path reconstruction errors by comparing the estimated positions $P^i(t)$ of walkers at timestamp t with the waypoint positions $\bar{P_j^i}$. It's important to note that an alignment process [67] is necessary before comparison due to the undefined reference frame of an estimated trajectory. To address this, we employ Procrustes analysis [30] to determine the rotation and translation that minimizes the squared distance between $\bar{P_j^i}$ and $P^i(t_j^i)$.

Three metrics are used to evaluate the estimated trajectories. The first metric is the Root-mean-square deviation (RMSE), with N representing the number of waypoints:

$$\mathrm{RMSE_{wp}} = \sqrt{\frac{1}{N}\sum_j \left\| \bar{\mathbf{P}}_j - \mathbf{P}^i{}_j(t_j^i) \right\|^2} \tag{7.1}$$

The estimated trajectory can be further sampled into N_e^i points Q_m^i with an inter-sample distance of 1 meter. A similar procedure is applied to the approximated ground truth trajectory, which is generated by connecting consecutive waypoints. Consequently, the ground truth can be represented by N_{gt}^i points Q_n^i. Please note that the sampling algorithm was developed and implemented by another Ph.D. student in our Lab. With these two sets of points,

we can evaluate the model using the Haussdorff distance and average Haussdorff distance [62], allowing us to comprehensively assess the goodness of the estimated trajectory.

$$\text{Hauss} = \max\left(\max_m\left(\min_n\left(\|\mathbf{Q}_m^i - \overline{\mathbf{Q}}_n^i\|\right)\right), \max_n\left(\min_m\left(\|\mathbf{Q}_m^i - \overline{\mathbf{Q}}_n^i\|\right)\right)\right) \tag{7.2}$$

$$\text{avHauss} = \frac{1}{2}\left(\sqrt{\frac{1}{N_e}\sum_m \min_n\left(\|\mathbf{Q}_m^i - \overline{\mathbf{Q}}_n^i\|^2\right)} + \sqrt{\frac{1}{N_{gt}}\sum_n \min_m\left(\|\mathbf{Q}_m^i - \overline{\mathbf{Q}}_n^i\|^2\right)}\right) \tag{7.3}$$

7.3.2 Map-less Path Reconstruction

In this section, we discuss the utilization of the following algorithms for reconstructing trajectories without prior knowledge of the building map (and therefore without employing a particle filter), as depicted in Figure 7.1:

1. **Azimuth/Steps (A/S):** In this approach, azimuth data is employed to determine the user's orientation, and an LSTM-based step counter developed by another Ph.D. student in our group is used for step counting. A PDR system with a fixed step length is applied to reconstruct the paths.

2. **45° – 90° Turns/Steps (T/S):** This algorithm is similar to the first one, with the distinction that the user's orientation is provided by the two-stage 45° or 90° turn detector.

3. **RoNIN (R) – Fine-tuned RoNIN (FR):** An introduction to RoNIN can be found in subsection 2.2.3. To evaluate the WeAllWalk dataset using RoNIN, the sensor data are upsampled from 25Hz to 200Hz through linear interpolation. I utilized the PyTorch implementation of RoNIN based on ResNet18 [34], which is provided by the authors [1]. For fine-tuned RoNIN, the model is trained further using data from blind walkers in the

[1] https://github.com/Sachini/ronin

WeAllWalk dataset. This process involves updating RoNIN's weights with additional training data from visually impaired individuals. However, the WeAllWalk dataset only provides timestamps when users pass each waypoint, whereas RoNIN requires continuous location and orientation data for training. To address this, I assumed that users walk at a constant velocity within each corridor segment in the paths from WeAllWalk, and that their orientations are parallel with the corridors. This assumption allowed me to generate the required ground truth by interpolating locations between waypoints. The RoNIN network was fine-tuned for two epochs with a batch size of 128, using the Adam optimizer [37] with a learning rate set to 0.0001. Since RoNIN was originally trained on data from sighted individuals, I only used data from blind walkers for fine-tuning.

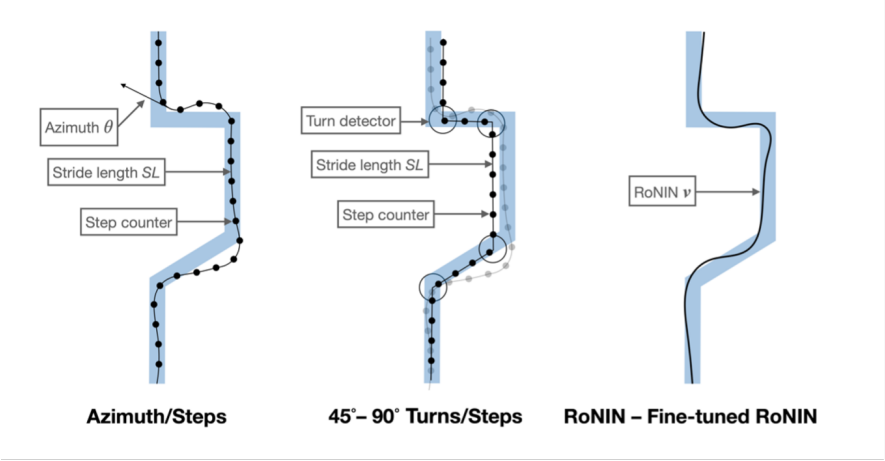


Figure 7.1: A visualization of three path reconstruction algorithms used in the map-less case is presented. The ground truth path taken by the walker is depicted in blue, while the estimated paths are shown in black. Heel strikes are represented by dots, and turns are represented by circles.

Results employing the three evaluation metrics are displayed in Table 7.2. According to this table, the combination of a 90° turn detector along with an LSTM-based step counter

	A/S			45° T/S			90° T/S			R			FR		
TS:LC	9.43	14.68	4.90	9.17	14.48	4.70	9.03	13.90	4.63	5.43	8.56	2.93	–	–	–
TS:DG	5.81	8.97	3.30	5.64	8.67	2.82	4.94	**7.50**	2.75	5.66	8.55	2.84	–	–	–
TC:S	4.45	7.06	2.39	3.96	6.12	1.89	**3.93**	**5.97**	**1.88**	4.26	6.75	2.39	–	–	–
TC:LC	3.85	6.21	2.30	3.86	6.45	2.22	**3.46**	**5.47**	**1.97**	5.43	8.56	2.93	4.36	7.37	2.54
TC:DG	6.38	9.90	3.29	6.28	9.86	2.92	6.13	9.60	2.92	5.66	8.55	2.84	6.80	10.42	3.29
TA:LC	6.29	10.27	3.60	5.99	9.79	3.32	5.88	9.47	3.31	5.43	8.56	2.93	6.18	9.90	3.27
TA:DG	5.21	8.37	2.88	5.00	8.18	2.52	**4.59**	7.64	**2.50**	5.66	8.55	2.84	5.17	8.08	**2.50**

Table 7.2: Reconstruction errors ($RMSE_{wp}$, Hauss, avHauss) of the path reconstruction algorithms applicable in the map-less case are presented, with units in meters. The smallest error values for each metric within each community of walkers (S, LC, DG) are displayed in boldface.

yields the best results across most modalities. Notably, for long cane users, models trained on data from sighted individuals (TS: LC) produce poorer performance compared to models trained on data from long cane users (TC: LC). Additionally, the performance of TC: LC is comparable to the best modality for sighted walkers, reflecting impressive results. For dog guide users, training models with all available data (TA: DG) consistently produces the best results in most cases. It's worth noting that fine-tuning RoNIN does not significantly improve the path reconstruction performance.

7.3.3 Map-assisted Path Reconstruction

In this scenario, we employed three different algorithms in conjunction with particle filter techniques, utilizing knowledge of the building maps. The selected algorithms include Azimuth/Steps (A/S), RoNIN, and fine-tuned RoNIN. Notably, the Turns/Steps algorithm was excluded as it demonstrated poor results in this context. We evaluated the performance of these

algorithms using three types of particle filters:

- **PF**: The standard particle filter.
- **PF-MS**: A particle filter enhanced with a mean-shift algorithm in the local maximum mode (as described in subsection 4.4.1).
- **PF-MS-G**: A particle filter enhanced with a mean-shift algorithm in the global maximum mode (as detailed in subsection 4.4.1).

The results of these evaluations are presented in Table 7.3.

	A/S			A/S PF-MS			A/S PF-MS-G			R PF			R PF-MS			R PF-MS-G			FR PF			FR PF-MS			FR PF-MS-G		
TS:LC	5.68	8.11	2.62	5.98	9.48	2.80	5.53	8.40	2.44	4.80	7.01	2.47	5.01	7.72	2.54	4.83	6.90	2.28	-	-	-	-	-	-	-	-	-
TS:DG	4.07	5.78	1.78	4.15	6.36	1.81	3.87	5.86	1.62	5.19	7.45	2.48	5.28	7.61	2.46	5.25	7.33	2.33	-	-	-	-	-	-	-	-	-
TC:S	3.35	4.93	1.60	3.35	5.01	1.48	3.32	4.80	1.38	3.10	4.46	1.46	3.29	4.92	1.45	**2.96**	**4.24**	**1.19**	-	-	-	-	-	-	-	-	-
TC:LC	2.78	3.95	1.36	2.86	4.35	1.28	**2.73**	**3.49**	**1.08**	5.05	7.44	2.54	5.17	7.98	2.51	4.95	7.37	2.30	3.54	5.43	1.89	3.62	6.15	1.86	3.50	5.24	1.66
TC:DG	5.77	8.12	2.46	6.09	8.50	2.61	6.08	8.39	2.35	5.56	7.98	2.74	6.08	8.65	2.92	5.47	7.64	2.39	6.10	8.68	3.03	6.19	9.62	2.97	6.16	8.59	2.77
TA:LC	3.30	4.71	1.62	3.46	6.21	1.68	3.20	5.06	1.44	5.36	7.69	2.54	5.55	8.38	2.62	5.40	7.69	2.40	4.53	6.53	2.32	4.79	7.29	2.42	4.51	6.55	2.13
TA:DG	4.19	6.10	1.99	4.62	6.44	2.03	**3.80**	**5.13**	**1.54**	5.09	7.56	2.48	5.29	8.13	2.46	5.23	7.62	2.40	4.28	6.01	2.16	4.32	6.64	2.14	3.97	5.56	1.81

Table 7.3: Reconstruction errors (RMSEwp, Hauss, avHauss) for map-assisted path reconstruction algorithms. Units are in meters. The smallest error values of each metric for each community of walkers (S, LC, DG) are shown in boldface.

Table 7.3 illustrates that a combination of the azimuth/steps (A/S) algorithm and the particle filter algorithm (PF-MS-G) yields the best results for the visually impaired community. Notably, the constraint of wall impenetrability significantly reduces drifts and path reconstruction errors. Consistent with the prior cases discussed, models trained on data from sighted individuals do not perform effectively on visually impaired individuals.

7.3.4 Results Visualization

Figure 7.2 presents visualizations of paths reconstructed by different algorithms in both map-less and map-assisted scenarios. In Figure 7.2 (a)(b), both algorithms (A/S, FR) produce trajectories with substantial drifts. The application of the particle filter successfully eliminates these drifts and reconstructs paths accurately. In Figure 7.2 (c)(d), the estimated path is either too short (A/S) or too long (FR). For the A/S case, the path can be correctly reconstructed using PF-MS-G. However, PF-MS-G struggles to rectify trajectory errors in the FR case. A similar scenario is observed in panel Figure 7.2 (e)(f), where PF-MS-G performs well in the A/S case but poorly in the fine-tuned RoNIN case.

7.4 Conclusion

In this chapter, we began by discussing the training and testing modalities, addressing them separately for long cane users and dog guide users. Three distinct schemes, denoted as TS, TC, and TA, are considered. We evaluated the two-stage turn detection algorithm using the Longest Common Subsequence (LCS) algorithm on the WeAllWalk dataset. This evaluation involved computing overcount and undercount values based on the matching results.

Furthermore, we examined three different methods for reconstructing the user's path in the WeAllWalk dataset. These methods included Azimuth/Steps (A/S), T/S (Turns/Steps), and RoNIN. We conducted evaluations in two cases: the map-less case and the map-assisted case. Notably, if a map was available, we integrated A/S, RoNIN and fine-tuned RoNIN methods with map information using a particle filter.

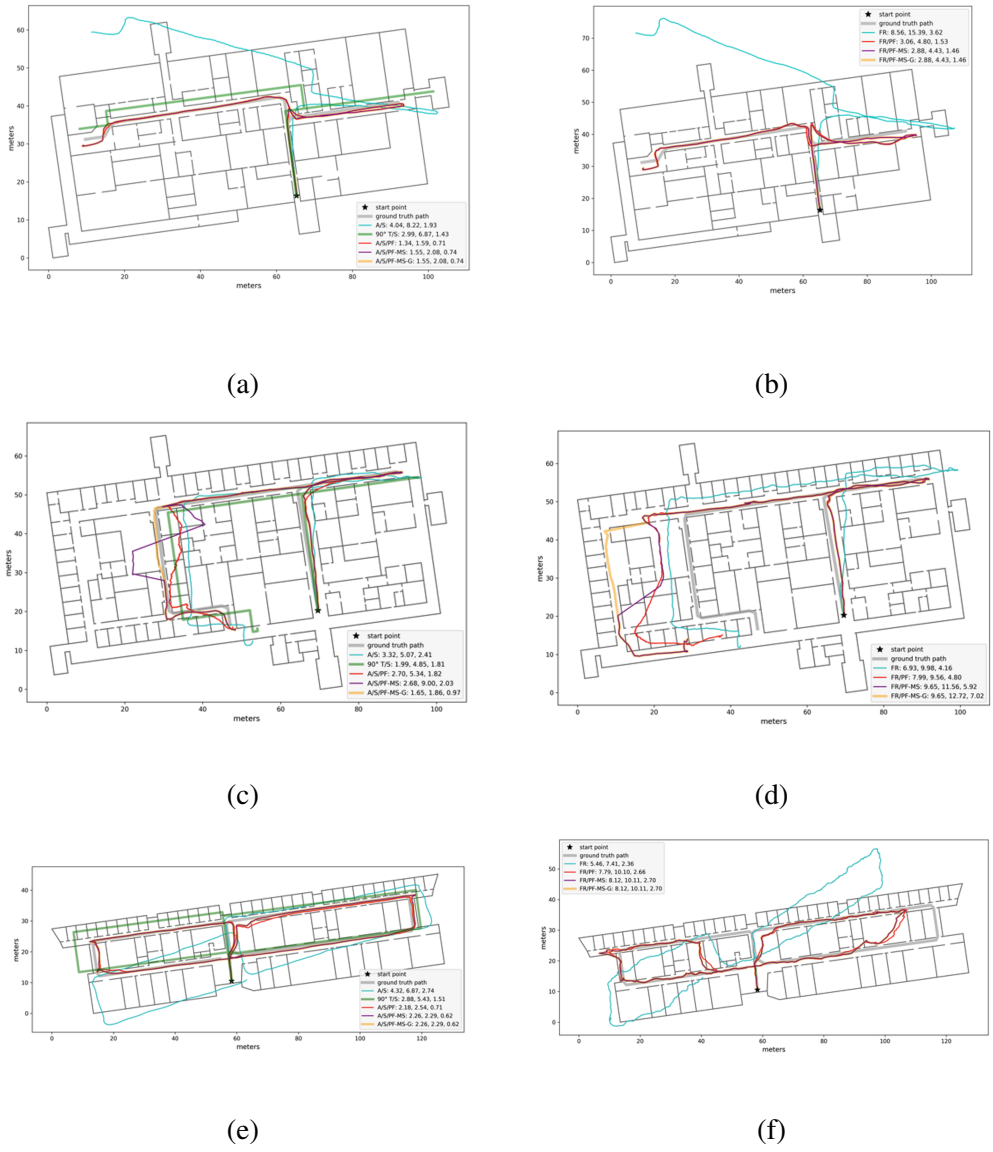


Figure 7.2: Path reconstruction examples from the TA: LC training/test modality. On the left: Map-less scenarios using A/S and 90° T/S algorithms; Map-assisted scenarios using A/S-PF, A/S-PF-MS, and A/S-PF-MS-G. On the right: Map-less scenario using the FR algorithm; Map-assisted scenarios using FR-PF, FR-PF-MS, and FR-PF-MS-G. The legend of each figure displays the values of three metrics (RMSEwp, Hauss, avHauss). All measurements are in meters.

To assess the accuracy of path reconstruction, we compared the reconstructed paths to the ground truth using three distinct metrics: Root Mean Square Error (RMSE), Haussdorff distance, and average Haussdorff distance. The visualization results illustrated that the particle filter effectively corrected the drift problem within the paths reconstructed by A/S or fine-tuned RoNIN.

Chapter 8

Experiments with Indoor Navigation

In this chapter, we will explore the practical implementation of the indoor navigation method previously discussed in Chapter 5. Navigating in the indoor scenario can be confusing for those who visit the building for the first time, and this becomes more challenging for visually impaired users due to the fact that they are not able to obtain visual information. Additionally, though multiple researches has been presented nowadays, there still doesn't exist a wide-spread commercially available system that can be used by the blind people. To guide a visually impaired individual from one location to another indoors, we have developed a navigation app for iOS. This application utilizes a particle filter and data from the Inertial Measurement Unit (IMU) sensor to ascertain the user's position and aid in navigation. We begin this chapter by delving into an indoor navigation app, covering its localization algorithm, interface design, and operational steps. Next, we examine the findings from a user study involving seven visually impaired participants, outlining the localization results of the app. Finally, we discuss the online service that I developed, which is capable of performing map format conversions.

8.1 Indoor Navigation Application Introduction

Let's first focus on a practical indoor navigation application designed specifically for visually impaired individuals. This application empowers users to determine a route from their current location to a selected destination, within indoor environments. It efficiently serves its navigational function by solely relying on the Inertial Measurement Unit (IMU) sensor and map data, thus allowing the phone to be conveniently carried in a pocket. We've already talked about the sensor fusion algorithm in this app in Chapter 5.

The app uses an Apple Watch and a Bluetooth bone-conduction earphone for its user interface. Users can control the app on their iPhone through the Apple Watch, using swipes and crown rotations. Then, the app's responses are sent to the Bluetooth earphone. This setup allows users to clearly hear the app's feedback through the bone-conduction earphone, even in noisy environments.

8.2 Localization Model Details

One of the most critical aspects of the navigation application is accurately localizing the user, as the design of navigation functions heavily depends on the accuracy of user localization. However, despite assuming a known starting point and initial walking direction, the localization task remains highly challenging due to the limited sensors available and the inherent drift in the IMU sensor. In this section, we will begin by discussing the two Pedestrian Dead Reckoning (PDR) methods utilized in the application, followed by an explanation of two essential calibration methods. Additionally, we will provide insights into the map details, which

are essential as they are utilized by the particle filter to handle the drift associated with the two PDR methods. Finally, we will discuss the various parameters used by the particle filter.

8.2.1 Utilizing IMU Sensor Data: Two PDR methods

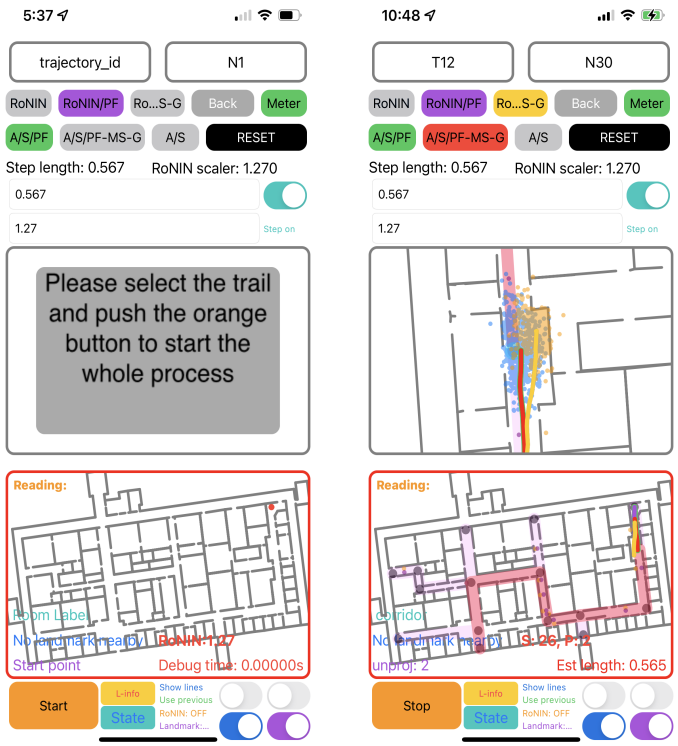
The application utilizes two Pedestrian Dead Reckoning (PDR) methods: Azimuth/Steps and RoNIN. Although only one method's results are presented to the user, the concurrent operation of both methods offers distinct advantages. Firstly, it serves as a fail-safe mechanism; in the rare event of one algorithm malfunctioning, the other can maintain operational continuity. Such an occurrence was observed only once in our experiments. Secondly, running both methods concurrently enables real-time comparison of their performance under varying conditions, as visually depicted in Figure 8.1 (b).

8.2.2 Calibration Methods

To ensure the correct functioning of the application, two distinct types of calibration are essential. These calibrations are required to determine the stride length or scalar for the Pedestrian Dead Reckoning (PDR) methods and to align the reconstructed path with the correct world reference frame. Let's discuss each calibration process in detail:

8.2.2.1 Stride Length Calibration

For the Azimuth/Steps method, prior knowledge of the user's stride length is crucial and must be obtained before conducting any experiments. Although we have access to the WeAllWalk dataset, which allows us to estimate an average stride length and use it as a constant,



(a) (b)

Figure 8.1: The application's debug interface: (a) Initial Interface - Displaying trail and target information, buttons for modifying display modes, settings for calibrated step length and RoNIN scaler, as well as options for enabling step beeps. It comprises two visualization screens with distinct scales, along with controls for app initiation/termination and various debugging functions. (b) Interface After Startup - Depicting two trajectories: one generated by the Particle filter + RoNIN (yellow line, yellow particles), and the other by the Particle filter + Azimuth/Steps (red line, blue particles).

practical experimentation has revealed that this approach lacks the required accuracy. Hence, we introduce a calibration phase for each participant. During this phase, participants traverse a known distance, and their stride counts are measured. Subsequently, the user's stride length can be easily calculated by dividing the known distance by the total stride counts.

Similar considerations apply to RoNIN. This model operates differently for various users with distinct scalars. Consequently, calibration of the RoNIN scalar is also incorporated into this process. Specifically, the known length is divided by the RoNIN-estimated length to determine the scalar value. During experiments, this scalar is used to adjust the RoNIN velocity.

It is worth noting that the stride length obtained through this calibration may still not yield optimal results. Visually impaired individuals may exhibit different walking patterns during experiments compared to the calibration path, as the latter is a narrow straight line, while experiment paths involve more complex features, including turns, open spaces, and landmarks. Therefore, additional adjustments are necessary. As previously mentioned in Chapter 5, multiple stride lengths are generated by introducing Gaussian noise to the calibrated stride length value. The particle filter then selects the appropriate stride length based on the map information.

8.2.2.2 Reference Frame Calibration

While the above-mentioned calibration procedure is time-consuming, it is only required once for each user across multiple trajectories. However, another vital calibration, known as reference frame calibration, is necessary for each trajectory. This is attributed to the utilization of two PDR methods, resulting in reconstructed trails within a frame in which the z-axis points downward and the orientation of the xy-axis is unknown. Aligning this reference frame

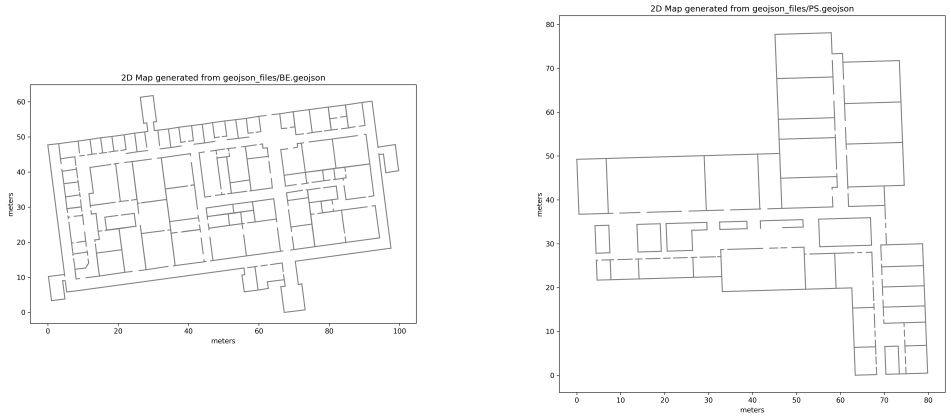
with the world frame is essential to accurately determine the user's position on the map. To achieve this alignment, we follow a specific procedure:

First, we collect a short traversed trajectory by instructing the user to walk forward for six steps in a predefined initial direction. We then reconstruct the path based on these initial steps and compare it to the known initial segment of the ground truth path to determine the angle required to align the two aforementioned frames: the frame of the reconstructed path and the world reference frame, which is used throughout the entire trail after calibration.

8.2.3 Map Information

Following calibration, the map serves as an strong constraint of information to correct errors in the navigation process. The map, initially provided in geojson format, has been converted into a suitable format using the online service mentioned in section 8.5. For our experimental purposes, we consider three maps, which originate from different locations within our campus: Engineering 2, Floor 3; Baskin Engineering, Floor 2; and the Physical Science Building, Floor 2. Their visualizations are depicted below:

Figure 8.2: Map visualization. Engineering 2, Floor 3



(a) (b)

Figure 8.3: Map visualization. (a) Baskin Engineering, Floor 2 (b) Physical Science Building, Floor 2

8.2.4 Sensor Fusion Algorithm

To leverage the map and enhance navigation accuracy, we employ the sensor fusion algorithm as previously discussed in the Chapter 5. In this section, we focus on essential parameters specific to the indoor scenario, omitting details covered elsewhere.

In our setup, we utilize 500 particles to balance computational speed and tracking accuracy effectively. During initialization, all particles start from a known location. The drift angle for each particle is sampled from a Gaussian distribution with $\sigma = 30^{\circ}$ and a zero mean. The step lengths are sampled from a Gaussian distribution with a σ value of 6 cm. It is important to note that all particles have equal initial weights, each being 1/500.

Upon detecting every step, we update the drift angle $\Delta_{\theta,i}$ of each particle using Gaus-

sian noise with $\sigma = 1^\circ$. Additionally, for particle i with stride length s_i, Gaussian noises are added to the velocity magnitude and velocity direction, with σ values of $\sigma = 0.5 \cdot s_i$ and $\sigma = 1^\circ$, respectively. During the resampling step that happens every step, a new particle is sampled using the available "healthy" particle, with a Gaussian noise of $\sigma = 10$ cm added to the position of the new particle. Finally, we compute the weighted mean as the resulting position.

It is crucial to emphasize that when employing PDR, the use of a particle filter to estimate the stride length is paramount. Otherwise, the calibrated stride length may not provide the required accuracy, particularly when the user needs to identify a turn between two walls, such as a T-shaped intersection. A failure case, where the particle filter is not used to estimate stride length, is illustrated in Figure 8.7(b), and it will be discussed in greater detail later.

8.3 User Interface Design

Despite users keeping their phone in their pocket, they still need notifications and guidance from the application. The app communicates with the user through a Bluetooth bone-conduction earphone and an Apple Watch, which provides haptic feedback via vibrations when crucial notifications, such as upcoming turns, are available. This feature proves especially valuable in noisy environments where information from the Bluetooth bone-conduction earphone may be challenging to distinguish.

Furthermore, users need to interact with the application. They may want to select a route, start or end the application, or revisit a previous notification in case they missed it. All of these interactions can be managed through the Apple Watch user interface. The following

gestures are available for the watch:

Select Route: Prior to starting a trail, participants are prompted to choose a specific route from a list. They can navigate the path list by swiping left or right in both directions. It is noteworthy that the path names are converted to participant via the Bluetooth bone-conduction earphone.

Start/End the Trail: Participants can initiate or terminate the entire trail process by rotating the watch's crown. Activation occurs when the user rotates the crown for two full rounds, after which they will hear a 'ding' sound from the watch. Depending on the app's status, they will either hear "Please start walking" if they are at the beginning stage of the app or nothing if they are ending the app tracking.

Information During the Trail: While on a trail, participants can swipe right to replay the last notification or swipe left to receive remaining route information from their current location to the destination.

Additionally, the app itself, although invisible and undetectable to users as it remains in their pocket, provides an interface on the screen for debugging purposes. While users walk, the app's content is transmitted to another iPhone for real-time monitoring, which is vital for assessing the model's condition (Figure 8.1 (b)). The app interface is explained in the caption of Figure 8.1.

8.4 Experimental Results

In this section, we start by offering information on the experimental procedure and a user study. Subsequently, we concentrate on the results of the experiments, with a primary focus on the localization outcomes achieved through the two methods, Azimuth/Steps and RoNIN. We will discuss their respective performance, considering both positive and negative examples.

8.4.1 Experimental Procedure

The experimental procedure for the application is outlined as follows:

1. The experimenter initiates the calibration process by clicking the calibration button within the app.

2. The user performs a one-time stride length calibration by walking a certain length of a straight line. During this process, the necessary scalar values are automatically calculated and recorded.

3. After calibration, the user is directed to the starting point of a trail. They can position the phone anywhere on their body while wearing an Apple Watch and a bone-conduction earphone.

4. Using the Apple Watch, the user selects their desired destination.

5. The user starts by walking straight for six steps, after which they receive essential notifications via the Apple Watch to guide them to their destination. Instructions may be repeated via the Apple Watch if needed.

6. Once the user reaches the destination, they can rotate the crown on the Apple Watch to stop the entire process.

8.4.2 User Study

All participants in this study are blind, and their information is presented in Table 8.1. Out of the seven users, four are female, and three are male. Additionally, two users prefer dogs, while five use canes. Their ages range from 53 to 76, with an average age of 70. The calibrated step length and RoNIN multiplier, along with the step length estimated by the particle filter, are also visible in this table.

Participant	Gender	Age	Blindness Onset	Mobility Aid	Step Length (Calibration)	Step Length (Final)	RoNIN Multiplier	Preferred Units
P1	F	73	L	Dog	48	–	0.96	Steps
P2	M	69	B	Cane	51	–	1.08	Feet
P3	M	53	B	Cane	54	–	1.14	Feet
P4	F	69	B	Cane	51	44	1.0	Feet
P5	M	75	L	Cane	44	41	1.21	Meters
P6	F	76	L	Cane	40	40	1.11	Steps
P7	F	72	L	Dog	63	58	1.08	Feet

Table 8.1: Participant Characteristics: 'B' denotes blindness since birth, 'L' indicates blindness later in life. The step length (Calibration) represents the estimated step length value obtained after the calibration phase for each user. The step length (Final) represents the average estimated step length value derived from particle data over three trials. It is worth noting that the Particle Filter estimates the step length as a state starting from participant P4. The preferred units indicate the type of units the user wishes to hear in the notifications.

8.4.3 Experiment Details and Localization Results

Each user's total experiment consists of four trials. To ensure user familiarity with the interface before the actual experiment, we include a practice trail for each participant. This

practice trail is conducted within the E2 building of our campus, which features multiple turns and landmarks, including benches along the route (Figure 8.4).

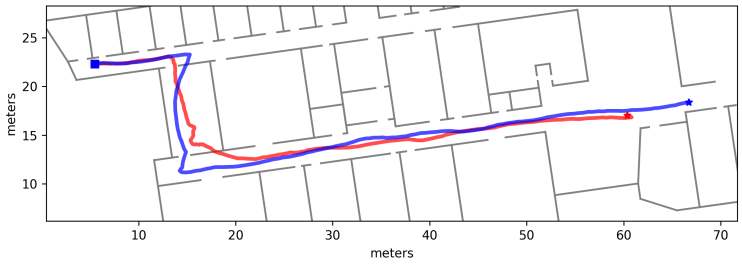


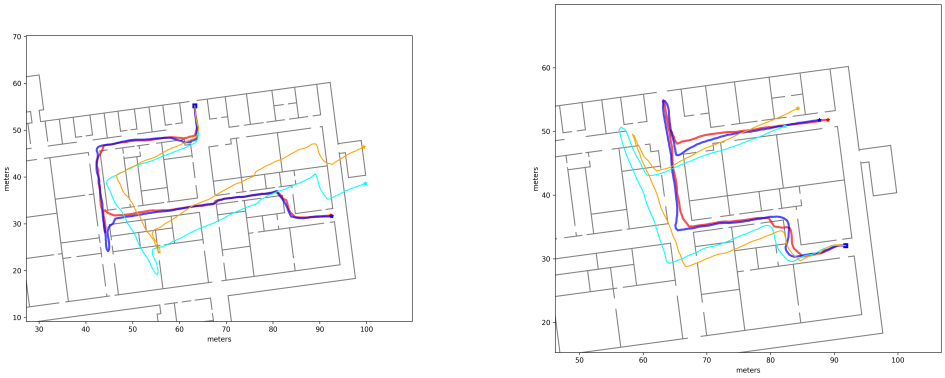
Figure 8.4: A practice trail within the E2 building from participant 2. Red line: Particle filter + RoNIN, Blue line: Particle filter + Azimuth/Steps. Start Waypoint: Square, End Waypoint: Star

After completing the practice trail, we guide the user to the Jack Baskin Engineering building, where three trajectories are traversed consecutively by all participants: R1W, R2W, and R3W. Visualization of these trails can be found in Figure 8.7, Figure 8.8, and Figure 8.9. Each trajectory's endpoint serves as the starting point for the subsequent trajectory. Please note that we did not test the physical science building because most participants ran out of energy after completing the practice trail and the three trials on the Jack Baskin Engineering building. The comprehensive details regarding all three trajectories are as follows:

Overall, the system functions efficiently. In Figure 8.5, one can see that the drift error within either RoNIN or Azimuth/Steps gets removed significantly by the sensor fusion algorithm. With occasional intervention from the experimenter, the app successfully navigates users from the start point to their destination, even under challenging conditions.

	P1	P2	P3	P4	P5	P6	P7	Length
R1W	261 (R,E)	355 (R)	297 (R)	216	271	223	206	123 m
R2W	304 (E)	209	134	163	211	262	171	97 m
R3W	125	170	98	127	144	139	330	72 m

Table 8.2: This table summarizes the results of the Wayfinding routes experiment. Completion times for every route is reported in seconds. If a participant missed some turns or took a wrong turn but completed the route with app guidance, the table cell will be displayed with a gray background. Here, 'R' indicates that a system reset was required, and 'E' indicates that, while the route was completed, verbal instructions from the experimenter were needed during the trail.



(a) (b)

Figure 8.5: Reconstructed paths from different modalities. Red line: Particle filter + RoNIN. Blue line: Particle filter + Azimuth/Steps. Cyan line: Azimuth/Steps. Orange line: RoNIN. Start Waypoint: Square, End Waypoint: Star. (a) Trail R2W for Participant P2. (b) Trail R3W for Participant P2.

8.4.4 Challenges and Issues: Discussion

Now, let's examine three scenarios in which the application exhibits less than perfect performance. These situations include: "Switching operation," where manual switching from Azimuth/Steps to RoNIN model is necessary; "Extra instructions," where the experimenter provides additional instructions alongside the app's guidance; and "Missing turns," in which case the user misses a turn but recovers to the correct one under the instruction of the app.

8.4.4.1 Switching Operation

As indicated in Table 8.1, there are situations where a reset or switch is necessary, marked as 'R.' In the case of P1, we had to switch to RoNIN after the first turn. This was due to changes in the user's walking pattern near a staircase, resulting in a significantly shorter stride length compared to the constant stride length obtained during the calibration phase. Consequently, when the user made a turn, Azimuth/Steps underestimated the user's trajectory, resulting in all particles colliding with the wall. The trajectory is depicted in Figure 8.7 (b). Thus, the transition to RoNIN became necessary.

For Participants P2 (Trail R1W) and P3 (Trail R1W), a reset was required for similar reasons. The constant stride length didn't meet the necessary accuracy needed for indoor navigation. Therefore, after P3, the stride length was estimated by the particle filter, and there was no longer a requirement to switch from P4 to P7. However, it's worth noting that while we chose to switch to RoNIN in certain conditions, RoNIN isn't always guaranteed to provide accurate results. For example, in Figure 8.10 (a), RoNIN fails to track the user's position correctly during the first trail of P2, R1W. This observation suggests that running both PDR methods in

parallel remains advantageous in the current state of the application.

8.4.4.2 Extra Instructions

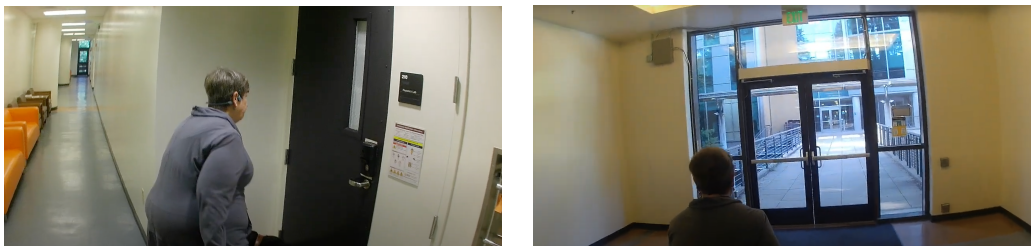
In rare cases, we found it necessary to provide extra instructions to the user. This occurred twice during our experiments:

The first instance involved P1, who was accompanied by a guide dog. During the experiment, P1 encountered difficulty in an alcove to the right (Figure 8.6 (a)). While she might have eventually determined the correct direction, we chose to provide her with instructions, leading the app to successfully guide her to the destination after the intervention.

The second case requiring intervention also involved P1 during the R2W trajectory. Initially, the instruction to turn right from the user to the guide dog was delayed because the user had already passed the intersection. Subsequently, when the user correctly turned around with app assistance and attempted to turn left following the instruction, the guide dog refused and continued walking straight toward the exit door. Thus, our intervention was necessary to prevent them from leaving the building. The dog's hesitation may have been due to nervousness related to the trails, as explained by P1.

8.4.4.3 Missing Turns

In the Table 8.2, cells shown in gray indicate instances where participants successfully completed the trajectories despite missing a turn. The app's notifications guided users back on track after the missing event. It's important to note that these cases may not be considered failures since participants were able to complete the entire trajectory with app assistance,



(a) (b)

Figure 8.6: (a) Participant P1 got stuck in the alcove on the right during Trail R1W of the experiment. (b) An intervention was provided to prevent the user from exiting the building during Trail R2W of the experiment.

without requiring any external help. Examples of these missing turn cases can be observed in Figure 8.7(d), Figure 8.8(c)(d), and Figure 8.9(c)(d).

The occurrence of missing turns can be attributed to various factors. For instance, participants might have been distracted by others nearby (P5 in R1W, Figure 8.10(b)) or engaged in conversation when notifications were issued (P2 in R2W, Figure 8.8(c)). Additionally, in large open spaces, users may struggle to identify intersections without adjacent walls (P6 in R3W, Figure 8.9(c)). The app's notifications may also have been slightly delayed, leading to a missed turn due to localization module undershooting. In our experiments (P6 in R2W, Figure 8.10(c)), the app promptly instructed the user to turn around, successfully guiding them through the turn.

An interesting case of a missed turn occurred when P1 (in R2W, Figure 8.8(d)) gave instructions received from the app to the guide dog, but the dog refused to obey and continued walking straight. Another factor contributing to missed turns is the user's walking speed. For instance, P7 missed a turn in R1W (Figure 8.7(d)) and R2W (Figure 8.10(d)) and multiple turns in R3W because of her fast pace with a guide dog. By the time notifications were completed

and delivered, the user had already passed the junction, resulting in a missed turn event. Notably, P7 also entered a small room through an open door at the beginning of one trail (R3W, Figure 8.9(d)) and missed the final turn of the same trajectory multiple times. Despite these outliers, the localization module effectively tracked the user, ensuring they reached the correct destination, highlighting its robustness.

8.5 Online GeoJSON File Processing Service

In Chapter 5, we established that while a GeoJSON file holds essential map data, it is not immediately usable by our model. Specifically, a parser script is required to extract essential details, including walls, landmarks, and room information, from the GeoJSON data. To render the map onto a 2D matrix for algorithmic application, we employ the Bresenham algorithm. To facilitate this, several Python3 scripts have been crafted. While these scripts are fully functional, they present a challenge: processing each new building's GeoJSON file is boring, demanding custom adjustments for every file, and manual integration of the resulting map into the app. To streamline this process, I introduced an online service. This service provides a user-friendly front-end for interaction and a back-end that handles file processing and serves APIs, details of which will be presented in the subsequent section. Although this service is straightforward, it significantly automates the inclusion of new buildings and accelerates the expansion of our app's navigational capabilities.

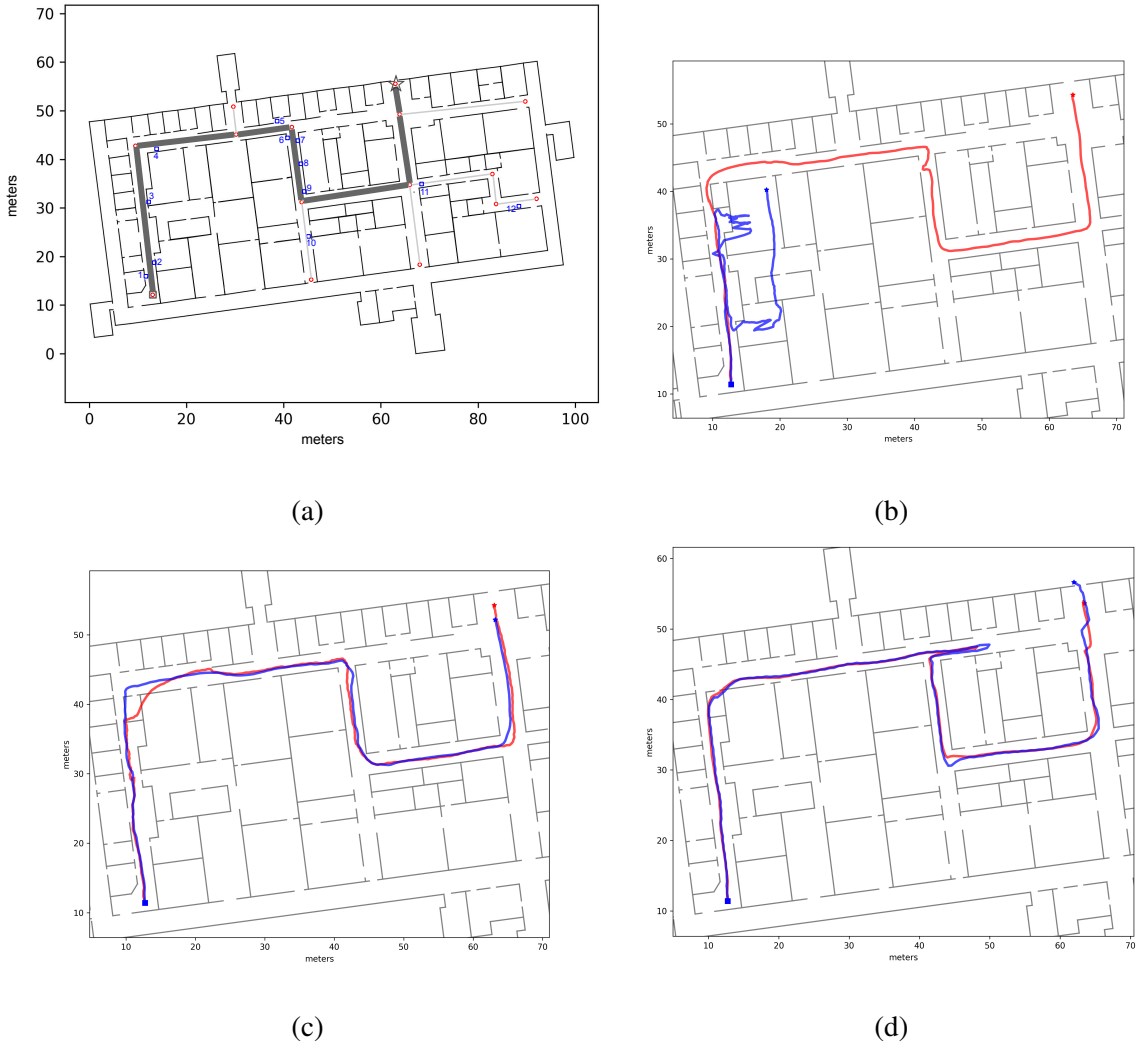


Figure 8.7: Route R1W. (a): Building floor plan with key elements highlighted: Red circles represent waypoints, Gray lines indicate traversable graph edges, Dark gray line shows the shortest path from the start waypoint (Square) to the end waypoint (Star). Landmarks are shown in Blue. (b)–(d): Recorded paths with Blue lines representing Azimuth/Step + Particle Filter and Red lines indicating RoNIN + Particle Filter. (b): Path for P1. (c): Path for P4. (d): Path for P7.

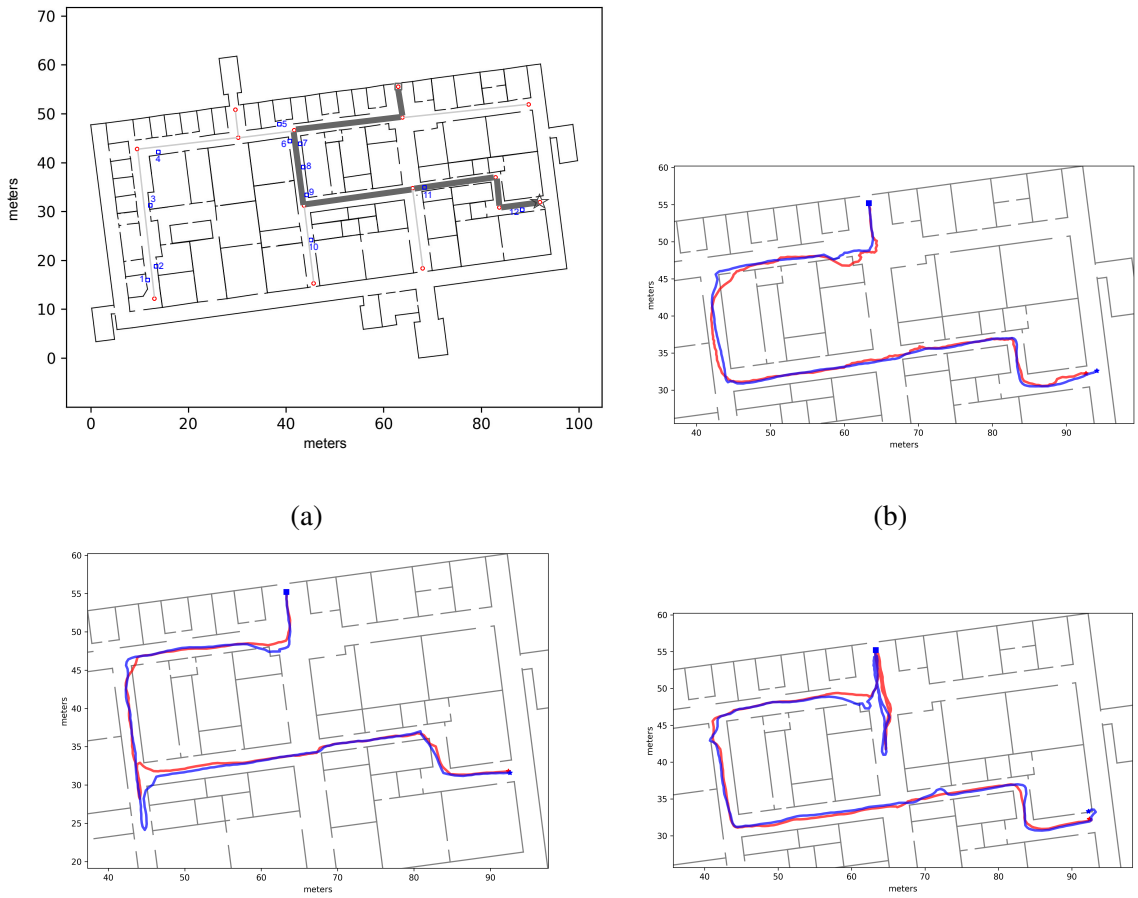


Figure 8.8: Route R2W. (a): Building floor plan with important features marked: Red circles denote waypoints, Gray lines represent traversable graph edges, Square indicates the start waypoint, Star signifies the end waypoint, Dark gray line connects the start and end waypoints. Landmarks shown in Blue. (b)–(d): Recorded paths with Blue lines showing Azimuth/Step + Particle Filter and Red lines indicating RoNIN + Particle Filter. (b): Path for P5. (c): Path for P2. (d): Path for P1.

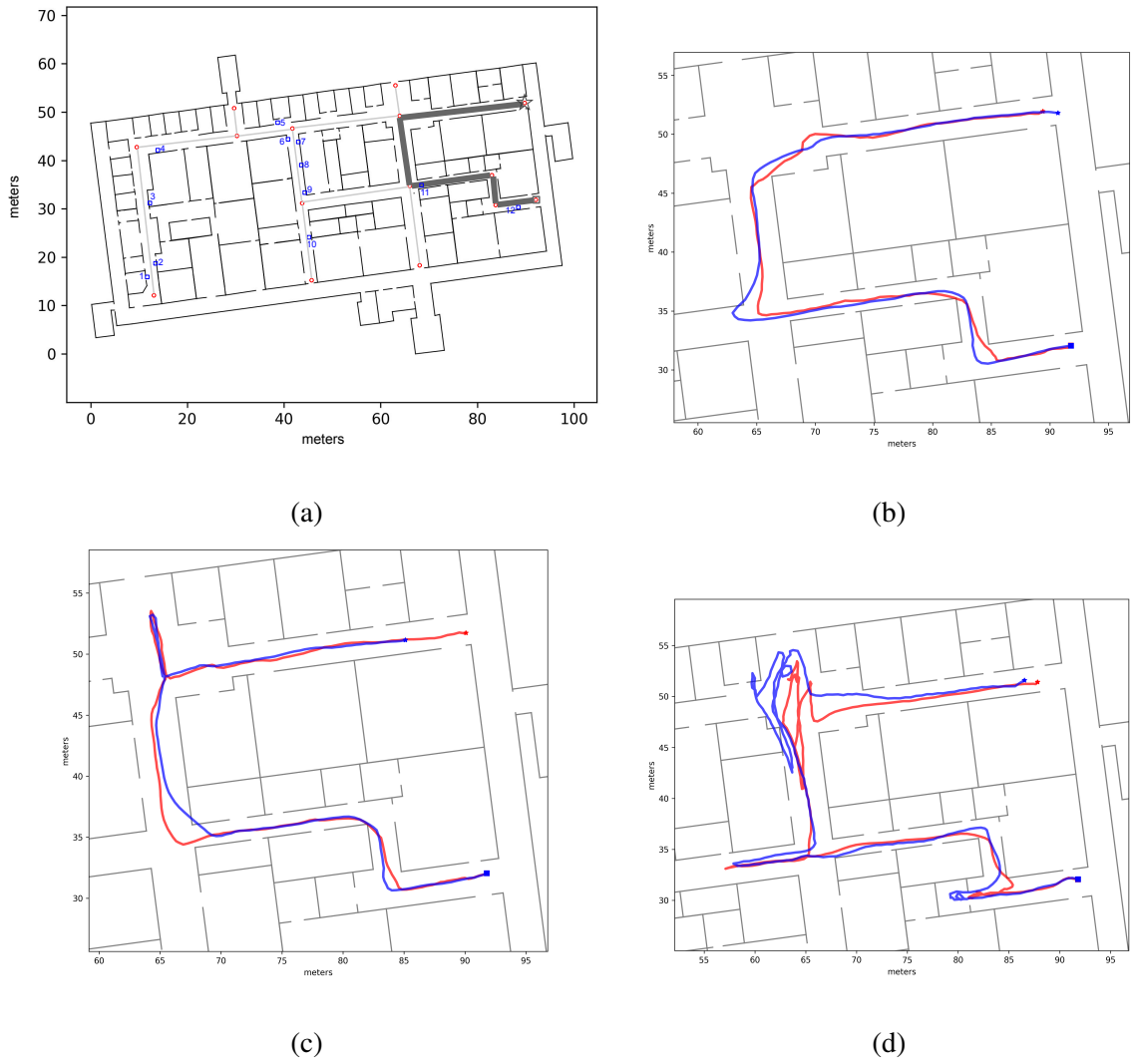
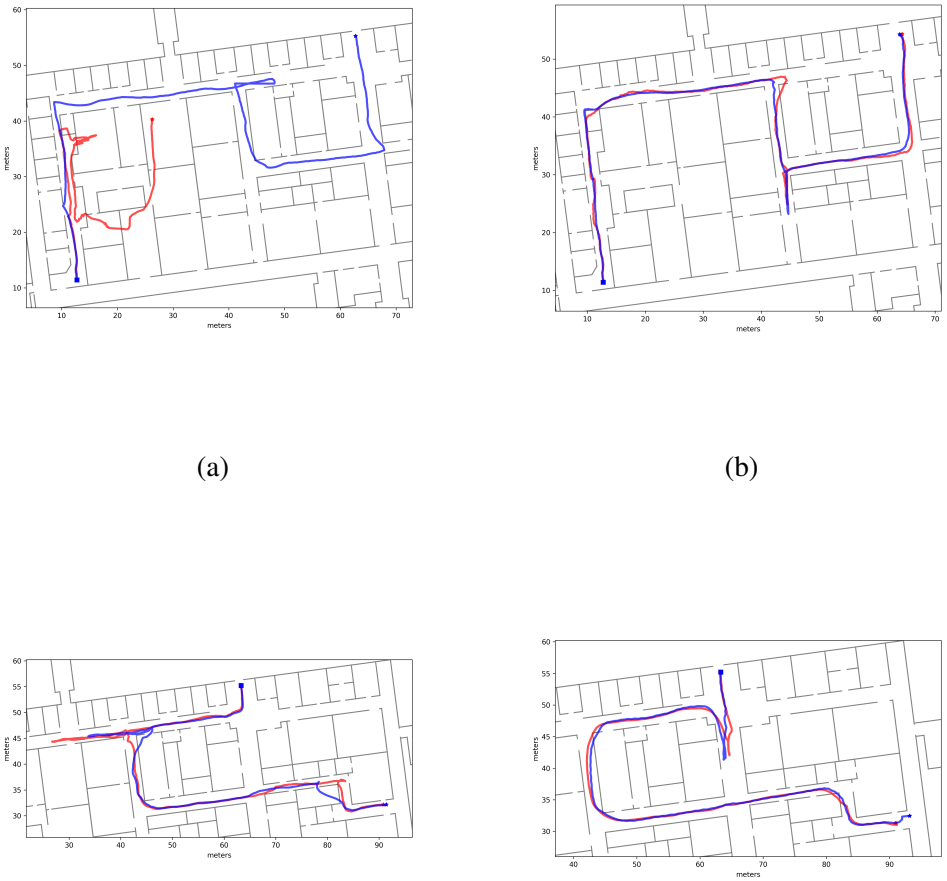


Figure 8.9: Route R3W. (a): Building floor plan highlighting significant elements: Red circles denote waypoints, Gray lines represent traversable graph edges, Square indicates the start waypoint, Star signifies the end waypoint, Dark gray line connects the start and end waypoints. Landmarks are shown in blue. (b)–(d): Recorded paths with Blue lines indicating Azimuth/Step + Particle Filter and Red lines indicating RoNIN + Particle Filter. (b): Path for P3. (c): Path for P6. (d): Path for P7. Note that P7 mistakenly enters a room via an open door at the early stage of the trail.



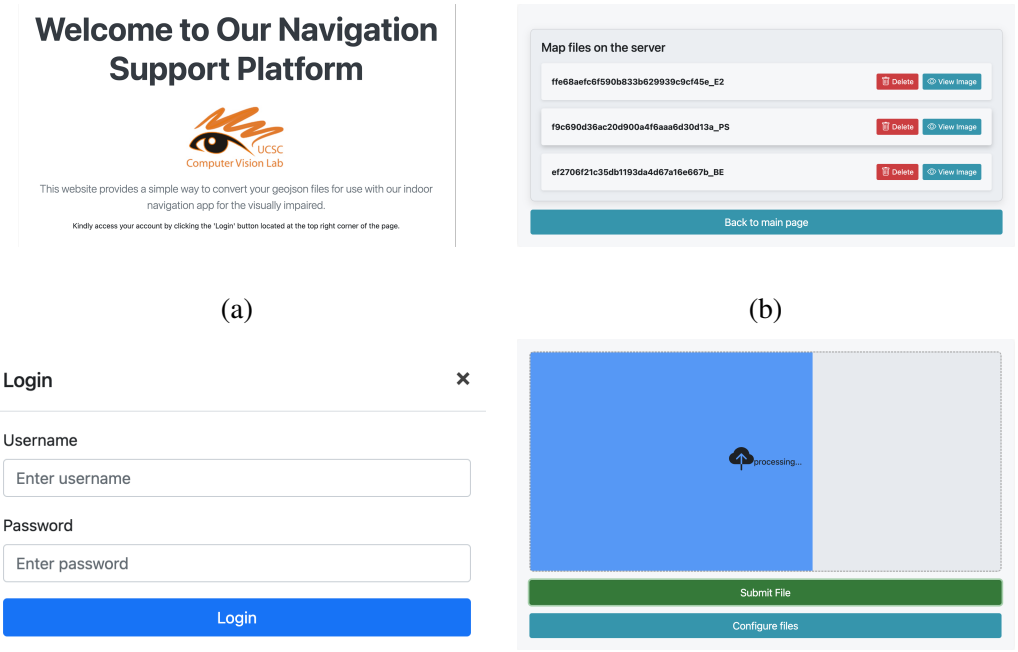
(a) (b)

(c) (d)

Figure 8.10: Localization results. Start Waypoint: Square, End Waypoint: Star. Red line: Particle filter + RoNIN, Blue line: Particle filter + Azimuth/Steps. (a) Route R1W, path for P1. (b) Route R1W, path for P5. (c) Route R2W, path for P6. (d) Route R2W, path for P7.

8.5.1 Front-end Website Introduction

The system's user interface is accessible via a website, developed using the widely-adopted VUE2 framework [1] and enhanced with the bootstrap-vue[2] package for an improved visual experience. It includes several key sections:



(a) (b)

(c) (d)

Figure 8.11: (a) Welcome Page. (b) Server File Management Interface for Remote File Configuration. (c) User Authentication Panel. (d) File Upload Page with Blue Progress Indicator.

1. **Welcome Page**: By navigating to **http://geojsonservice.online/** in a web browser, users are greeted by the welcome page (Figure 8.11 (a)).

2. **Login Panel**: To ensure security and prevent unauthorized access, a login feature restricts file submission to authenticated users only. Credentials are required for login, after which

[1] https://v2.vuejs.org/
[2] https://bootstrap-vue.org/

Figure 8.12: Data Visualization Page Displaying Processed Map Data Graphically.

the user receives a temporary bearer authorization key. This key must accompany all subsequent requests to the backend, thereby activating the service (Figure 8.11 (c)).

3. **Upload Page**: After logging in, users are redirected to the upload page. Users have the option to either select a GeoJSON file for upload via a dialog box or drag and drop the file directly onto the page. Once the upload process begins, the front-end continuously checks the backend's processing status. The completion time for processing varies, typically ranging from one to two minutes on a standard AWS EC2 t2.micro instance[3], but may decrease significantly with more powerful servers. A progress bar informs users of the upload status, enhancing the user experience (Figure 8.11 (d)). After processing, users can immediately review the parsed results and landmark positions directly on the site to confirm correct file handling by the backend (Figure 8.12).

4. **Control Panel**: The backend hosts the processed files as static assets, which can be

[3] https://aws.amazon.com/ec2/instance-types/t2/?nc1=h_ls

managed through the front-end interface. Users can configure which GeoJSON files are served by the server, view processed map images, or delete files as needed (Figure 8.11 (b)).

8.5.2 Back-end API Introduction

The backend operates on an AWS EC2 Ubuntu instance and utilizes Flask [4] in conjunction with Nginx[5] and uWSGI[6] to deliver the APIs discussed earlier. In this setup, uWSGI is employed to run a Flask application, while Nginx takes on the role of managing incoming requests from the frontend and serving static files. The APIs provided are as follows:

1. **Login API**: This API verifies the user's username and password against a server-stored file containing all registered accounts. Due to the experimental nature of the project and potential security issues, a registration API has not been implemented at this stage. Currently, only individuals with server access are able to modify the account file.

2. **Process API**: This API takes GeoJSON files uploaded from the front-end and processes them asynchronously into multiple required formats. Upon completion, the processed files are stored in a directory which Nginx serves as static content.

3. **Status API**: This API verifies the presence of specific files and returns a boolean value to indicate their existence. It is employed to assess the current task's status by checking for the presence or absence of the resulting file. While the current system does not use

[4] https://flask.palletsprojects.com/en/3.0.x/
[5] https://www.nginx.com/
[6] https://uwsgi-docs.readthedocs.io/en/latest/

a database for file management since the number of maps to be converted are limited, incorporating a database such as MySQL or MongoDB could be a future enhancement.

4. **File List API**: This API provides a list of all static map files available for download by our application.

5. **Delete API**: Through this API, users can delete specific map files from the server, enabling the remote management of files via the front-end.

8.6 Conclusion

This chapter primarily centers on the indoor navigation application. We begin by delving into the details of our localization model, with a particular emphasis on two crucial calibration processes: one for estimating step length and the other for reference frame calibration. These processes play a crucial role in ensuring the app's robust functionality.

Furthermore, we provide details about the map used in our study, coupled with the sensor fusion algorithm, grounded in the particle filter methodology. We also highlight the user-friendly design of our interface, tailored for Apple Watch.

Subsequently, we provide a brief overview of the experimental setup, including experimental procedure, participant information, and presenting detailed results. Our primary focus remains on the localization aspect. We carefully examine the outcomes of the application evaluation, analyzing both successful cases and failure cases.

Lastly, we introduce a straightforward and user-friendly online service that enables the seamless conversion of geojson files into the format needed by the navigation app.

Chapter 9

Experiments with Transit Hub Navigation

Navigating an unfamiliar transit hub can be a challenging and stressful task, particularly for individuals with visual impairments who lack access to essential wayfinding information through visual signage. In this chapter, we focus on the Transit Hub Navigation experiment, conducted through the utilization of an iOS application named RouteNav. We start by introducing various aspects of the application, namely, we will cover the localization model, fail-safe mechanism, and user interface design. Then, we present the outcomes of our experiments, with a primary focus on the localization findings, along with the discussion of the results.

9.1 Outdoor Navigation Application Introduction

RouteNav is designed to assist visually impaired individuals in navigating a transit hub that includes both overground and underground environments. This application enables users to begin their journeys from pre-defined starting points and guides them to destinations they have chosen in advance. The localization model incorporates multiple sensor components,

including Inertial Measurement Unit (IMU) sensors and GPS. These sensor data are fused with map data using a particle filter algorithm, as discussed in Chapter 6, to provide localization services.

The user interface has been carefully designed to assist users in navigating through challenging areas. To use the app, the user has the option to either hold the phone or place it in their pocket. Navigation instructions are delivered through a bone-conduction Bluetooth earphone, while interaction with the app is facilitated through VoiceOver[1] by interacting with buttons on the screen.

9.2 Localization Model Details

In this section, we focus on the details of the localization model. We begin by introducing the Palo Alto train station. Following that, we evaluate the performance when using only IMU or GPS data, and examine the map employed by our application. Lastly, we delve into the parameters of the sensor fusion algorithm, and talk about a fail-safe mechanism.

9.2.1 Transit Hub Information

In this experiment, our primary aim is to evaluate the performance of the application within the Palo Alto train station. The Palo Alto train station serves as a transportation hub, enabling a smooth transition between buses and trains. Notably, it features an underground tunnel that must be traversed by users seeking access to specific train boarding areas. The map

[1] https://developer.apple.com/documentation/accessibility/supporting_voiceover_in_your_app

of the Palo Alto train station is depicted in Figure 9.1.

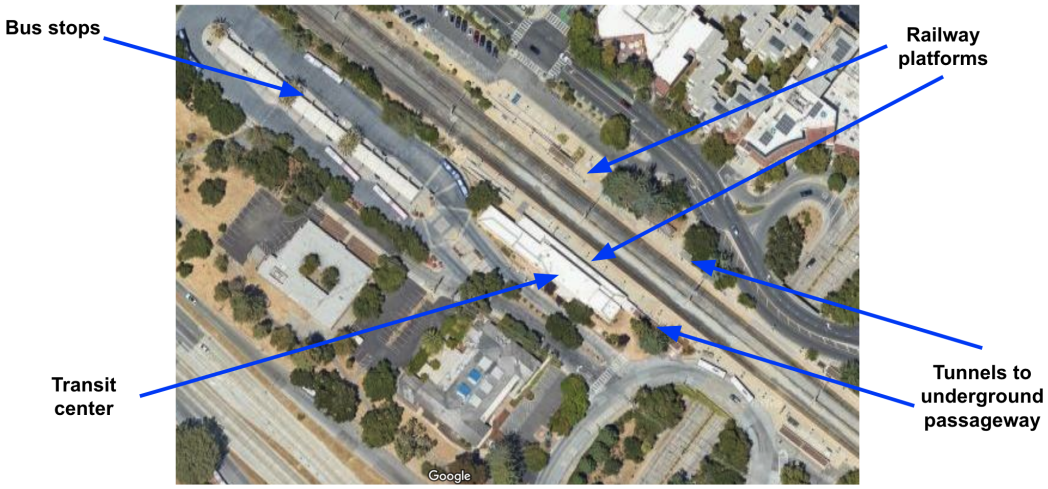


Figure 9.1: Map of the Palo Alto train station, highlighting the positions of bus stops, railway platforms, underground passageway tunnels, and the transit center.

9.2.2 Utilizing IMU Sensor Data

We first focus on the application of the RoNIN model as a key method for estimating the user's velocity and reconstructing their paths. Here we present two examples that highlight issues encountered with RoNIN during our experiments, as depicted in Figure 9.2. As we can see, in Figure 9.2 (a), evident drift is observed in the reconstructed path. In Figure 9.2 (b), the RoNIN velocity magnitude significantly deviates from the correct value, being excessively large. Additionally, these two trajectories in Figure 9.2 (a) and Figure 9.2 (b) demonstrate variations in RoNIN's performance among different blind users.

(a) (b)

Figure 9.2: Two RoNIN trajectories collected from different users on an overground map. White lines represent the paths traversed by the users, white circles represent the destinations, while the black lines represent the RoNIN reconstructed paths. (a) Bus2NB for participant 5. (b) Bus2NB for participant 1.

9.2.3 Map Information

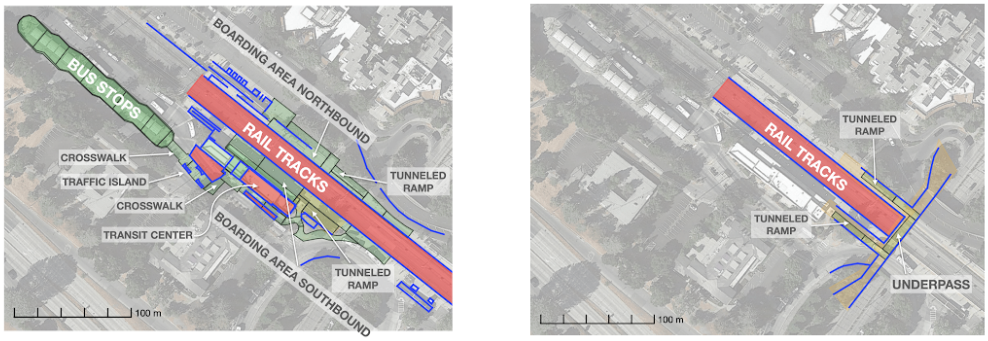
Since IMU sensor data alone is insufficient, map information is required to complement it. The following two figures, presented in Figure 9.3, depict the maps used in our system after post-processing: the overground map and the underground map. Various assumptions are associated with the different colored areas on these maps:

1. **Walls:** These walls are impassable barriers, and we do not expect users to traverse them.

2. **Walkable Zones:** Users are expected to walk within these areas.

3. **No-go Zones:** Users are discouraged from entering these areas, as they may be hazardous (e.g., railway tracks) or inaccessible (e.g., fenced zones).

4. **GPS-denied Zones:** These areas consistently exhibit significant GPS errors or unreliability. Consequently, we ignore GPS data when users are within these zones. For instance, in the overground map, this includes the area near the underground tunnel where the GPS signal is partially blocked. In the underground map, we do not use GPS since it's unavailable when users are within the underground tunnel.

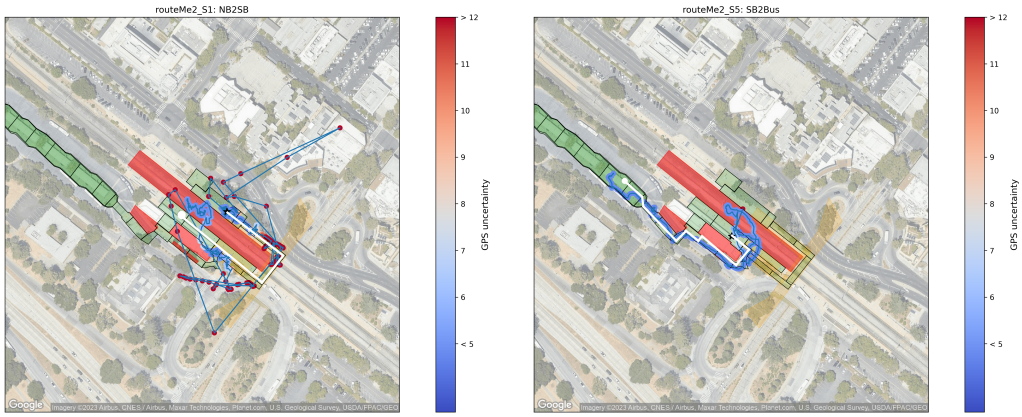


(a) (b)

Figure 9.3: Maps utilized by the Palo Alto train station navigation application. (a) Overground map, employed when users navigate above ground. (b) Underground map, utilized when users navigate within the underground tunnel. In both maps, distinct areas are differentiated by colors: blue lines represent impassable walls, green areas denote walkable zones, red areas signify no-go zones, and yellow areas indicate GPS-denied zones. Please note that the yellow areas are also walkable.

9.2.4 GPS Usage

In addition to map information, GPS, which is available outdoors, can also serve as a valuable source of data. Here, we initially focus on cases where only GPS is utilized. Unfortunately, as depicted in Figure 9.4 (a), GPS data can exhibit significant inaccuracies when the user walks underground. Additionally, when the user walks in the Southbound platform, the GPS has erroneous outcomes, despite relatively small uncertainties (Figure 9.4 (a)(b)).



(a) (b)

Figure 9.4: GPS outage examples. Blue line - GPS (uncertainty visualized by color), White line - Ground truth path, White Circle - Destination. (a) Participant S1, trail: Northbound to Southbound. (b) Participant S5, trail: Southbound to Bus Stop.

While relying solely on GPS or IMU data may not yield accurate results, the fusion of these signals with map data leads to improved outcomes. Now, let's delve into further details about the sensor fusion algorithm used.

9.2.5 Sensor Fusion Algorithm

At this stage, we employ a particle filter-based sensor fusion algorithm to merge data from the IMU sensor and GPS. The complete pipeline is illustrated in Figure 9.5. It is worth noting that the particle filter employed in this context shares similarities with the one used in the indoor navigation app mentioned in section 8.1. Specific differences in parameters are reported as follows: We utilize a total of 750 particles, each initialized with equal weights (1/750). The particle filter's prediction step operates at a frequency of 25 times per second. During each iteration, the drift angle $\Delta_{\theta,i}$ of each particle is updated by Gaussian noise with a sigma value

of $\sigma = 0.2°$. Notably, this value is smaller than that used in the indoor case, owing to the higher prediction frequency. Additionally, we introduce Gaussian noise to the velocity from RoNIN, with $\sigma = 4.3$ meters added to the magnitude and $\sigma = 4°$ added to the velocity direction. In the resampling step, each resampled particle is perturbed by Gaussian noise with $\sigma = 2.5$ meters. These larger noise values signify the more challenging outdoor environment compared to the indoor scenario.

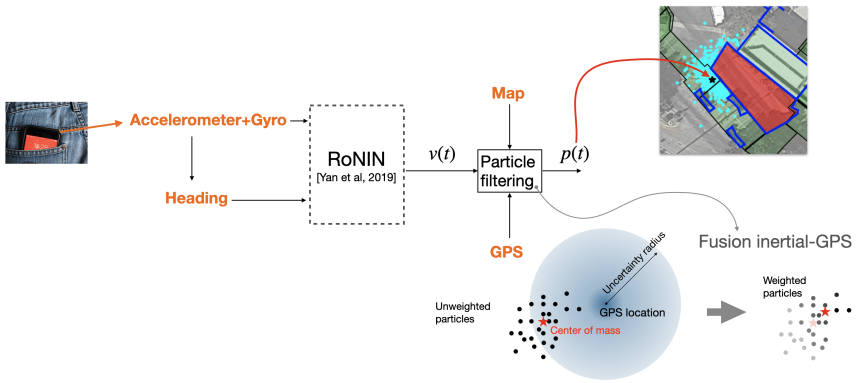


Figure 9.5: The sensor fusion model pipeline. RoNIN is employed to process the IMU sensor data collected from the iPhone in the user's pocket. Its output velocity is then used as an input to particle filtering, which incorporates the map information. For each particle, its weights are determined by the map and the GPS distribution. The weighted mean of a cluster of particles is used as the final output of the model.

9.2.6 Fail-safe Mechanism

While the entire system is efficient, it can still encounter failures in extreme cases. To address this, a fail-safe mechanism has been developed. Specifically, when the user exits a tunnel and the provided GPS position significantly deviates from the current particle filter prediction, if the GPS uncertainty remains small, we reset all particles to the GPS position.

Although this mechanism is effective in resolving errors within the particle filter as one can see in Figure 9.13 (a), it comes with some risks, especially when GPS uncertainty is unreliable. This problematic situation was observed in the experiment shown in Figure 9.13 (b), which will be discussed in more detail later. Hence, it is advisable to be cautious when employing this mechanism.

9.3 User Interface Design

Another crucial aspect of this application is the user interface. Unlike the previously mentioned indoor navigation app, we do not rely on an Apple Watch interface in this project. This project is earlier than the aforementioned one, and the use of an Apple Watch was not considered at this stage. Therefore, the primary means of user interaction with the app is through VoiceOver, accompanied by necessary feedback mechanisms such as vibration and sound when buttons on the screen are pressed. In this section, we will first introduce the user interface in general, then focus on the Route Compass, a highly valuable feature appreciated by multiple users.

9.3.1 User Interface

In this section, we will discuss the user interface design of RouteNav (see Figure 9.7 (b)). It is important to note that this is a collaborative project, and specific details related to the contributions of other collaborators have been omitted. We encourage the reader to refer to [56] for more information on their work.

Considering the limited accuracy of the localization model, we have taken great care in designing the user interface for RouteNav. To creatively illustrate spatial representation and traversal routes, we have introduced an innovative method using tiles, as shown in Figure 9.6. A tile is defined as an area of polygonal space, with a size that is at least equal to the expected localization resolution. The traveler's location, marked at each timestamp, is assigned to a specific tile (or an out-of-tile state), which contains detailed information such as highly localized features and landmarks. As a result, all information provided to the user is based on the tile that the traveler currently occupies. Furthermore, the use of tiles simplifies the representation of paths for users, making them easier to comprehend and remember.

In addition, to guide users to their target tile, we introduce a novel concept called the 'goalpost,' which replaces traditional waypoints that require physical arrival. Each tile is associated with two goalposts that serve as references for directional guidance. Importantly, a goalpost is used solely to set the travel direction, eliminating the need for the user to physically reach it. This approach mainly provides direction to the next tile and automatically shifts to a further one as the user approaches the current goalpost. This strategy, as illustrated in Figure 9.7 (a)(b), aims to reduce directional errors when the user's position lacks precision.

The RouteNav interface also incorporates redundant modalities to enhance efficiency. These modalities consist of two mechanisms for directing users to their target goalpost: user-solicited and user-unsolicited notifications. Both are illustrated in conjunction with various notifications offered by RouteNav in Figure 9.8. More specifically, for the user-solicited scenario, users can activate the interface by double-tapping the on-screen button to obtain a route preview or answers to the following questions through speech: "What's around me?" "Where

am I?". In the unsolicited case, users will receive speech alerts when they enter or approach a new tile. Furthermore, sound and vibration are utilized to assist the user in determining if they are moving in the correct direction. If the difference between the user's walking direction and the target direction exceeds 45°, the user will be notified by an unpleasant sound (nfc-scan-failure.caf) along with a brief vibration repeated twice every other second. Otherwise, a short whooshing sound (mail-sent.caf in iOS) will be generated, accompanied by a brief vibration every other second. Details about the Route Compass and Tunnel/Ramp alerts are provided in subsection 9.3.2 and section 9.4.

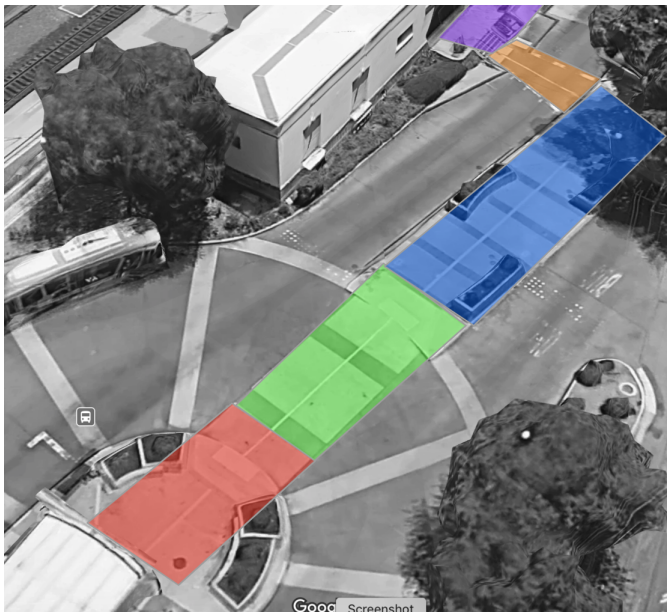
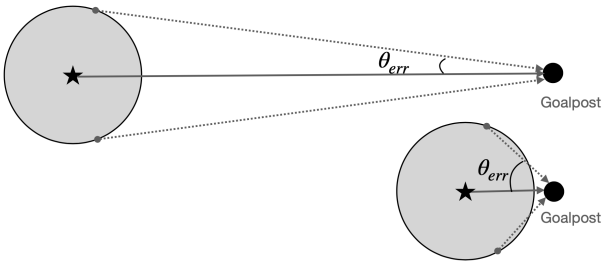
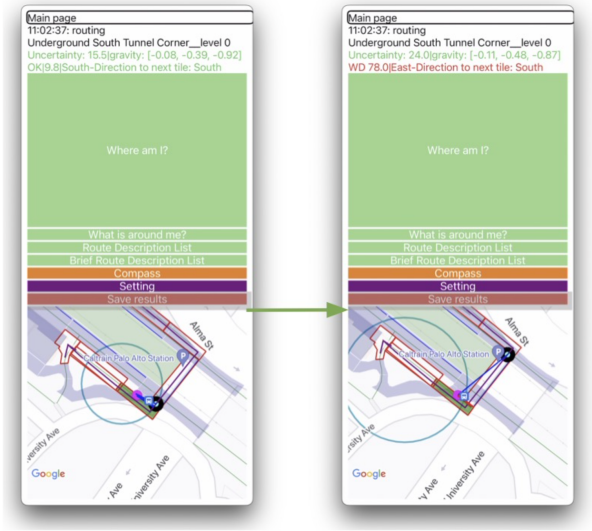


Figure 9.6: Visualization of Tiles on the map. Multiple tiles are displayed on the map, each with different colors.



(a)



(b)

Figure 9.7: (a) This schematic illustrates how directional error varies with distance to the next goalpost. The true location of the walker is marked by a star, but localization errors may place the user anywhere within the gray circle, which is determined by the uncertainty radius. Due to these errors, the system-generated direction to the target goalpost (thick dot), indicated by dotted arrows, may differ from the actual correct direction, represented by a solid arrow. The angle between these two directions forms the angular error, denoted as θ_{err}. The diagram shows that when the user is closer to the goalpost and has the same uncertainty radius, there is a higher likelihood of experiencing larger angular errors, represented by θ_{err}. (b) Screenshots of the RouteNav app. The screen displays multiple pieces of information, and various buttons are available for users to access information from the app by interacting with them using VoiceOver. The pink dot on the map represents the estimated position of the walker. The dark circle indicates the next selected goalpost. In the left image, as the walker enters a tile, the nearest goalpost in the following tile is chosen as the target. In the right image, as the walker advances within the tile, The target switches to the goalpost that is farther away in the next tile.

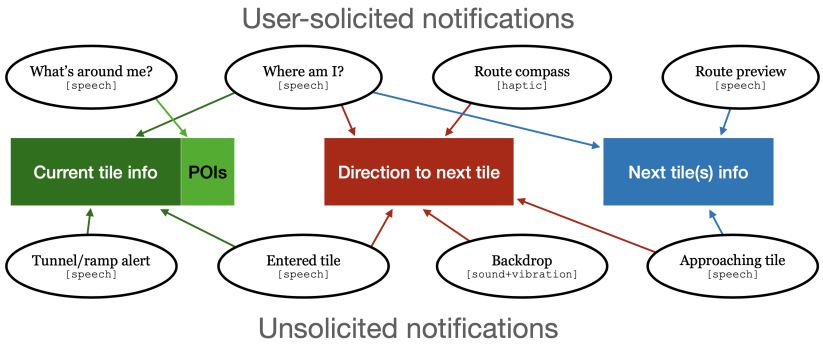


Figure 9.8: The RouteNav interface is illustrated using colored rectangles to represent distinct information categories offered by the application. A green rectangle displays details of the current tile, including points of interest (POIs). The red rectangle indicates the direction towards the target goalpost in the subsequent tile, while the blue rectangle provides information regarding the upcoming tiles on the route. Additionally, various notification types are represented by circular icons, each paired with its corresponding interface modality, which includes speech, sound, vibration, or haptic feedback.

9.3.2 Route Compass

During our experiments, we implemented a highly useful functionality known as the Route Compass, which resembles the ”wand” modality discussed by [7]. As the name suggests, the Route Compass serves as a directional tool, enabling users to determine the direction of their next target. In this feature, we leverage the magnetic compass built into the iPhone to provide absolute direction information. It's important to emphasize that in outdoor environments, the magnetic field tends to be significantly more stable than that in indoor environments, making compass-based orientation reliable. Specifically, we calculate the user's target direction and map it to the map, providing users with one of the following eight directions: East, West, South, North, Northeast, Northwest, Southeast, or Southwest. To use this feature, the user simply holds the phone in their hand and rotates it. A vibration feedback is triggered when the phone's direction aligns with their target direction. This feature is particularly helpful when users are

uncertain about their route and can aid them in mentally marking their destination. Figure 9.9 illustrates a user using the Route Compass during one of our experiments.

Figure 9.9: A participant using the Route Compass during one of the experiments, while holding a walking cane in the other hand.

9.4 User Interface for Challenging Spots

Despite the utilization of a particle filter-based localization model to merge various sensor data, localization accuracy can still pose challenges, particularly in spaces where open areas exist, and GPS signals are unreliable. In such scenarios, careful design of the user interface is essential to manage this inaccuracy. Here, we primarily focus on two challenging spots: the entrance of the underground tunnel and the ramp. We will first address the map-switching problem that arises near the entrance of the underground tunnel, followed by a similar solution for the ramp problem.

9.4.1 Challenging Spots: Underground Tunnel Entrance

As previously mentioned, the transit hub comprises two parts: the overground and underground sections. Users need to switch between these parts, and it's vital to switch maps at suitable timestamps. Ideally, with a highly accurate localization model, we could switch maps as soon as the user enters the underground tunnel. However, this is not always feasible, particularly given the frequent unreliability of GPS signals near the underground tunnel entrances. Furthermore, particles may gather in two groups near the entrance: one inside the tunnel and the other outside, causing unpredictability in predictions. Therefore, we need a better approach to address this issue. The Figure 9.10 shows one of the tunnel entrances.

Figure 9.10: An underground tunnel entrance pointed by a highlighted yellow line.

To facilitate map switching, one might consider using the altimeter from iPhone as an additional sensor. By utilizing the provided attitude data, we can track changes in the user's altitude and perform map switches based on these changes. However, altimeter data itself con-

tains biases that vary each time the app is started, making it unsuitable to set a fixed threshold for map switching.

To overcome this challenge, my second approach involves monitoring changes in altimeter data. If the altimeter data consistently increases or decreases, and the absolute change exceeds a certain threshold, we trigger the map switch. However, in the context of the Palo Alto transit hub's underground tunnel, which features a gentle slope, this method encounters a challenge. The altimeter data may change too slowly, causing a delay in the map switch. The solution to this problem involves treating the user as a sensor [21], which we will discuss in the next section.

9.4.2 User as a Sensor

The concept here is to empower the user to detect their surroundings and provide input to the app actively. In this approach, users are not passive recipients of instructions from the app. Rather, they are active agents capable of interacting with the app and the environment. This concept, referred to as "User as a Sensor" [21], acknowledges that users have greater access to environmental information than the sensors on a smartphone. For instance, users can interact with their surroundings by using their hands or a walking cane, receive guidance from a trained guide dog capable of perceiving visual information and navigating, or even seek assistance from sighted individuals nearby. As a result, input from these "sensors" is exceptionally reliable, and when designing the app, confidence in this "sensor" input should be set to a high value. Furthermore, respecting user input is essential for ensuring a positive user experience.

For the map-switching problem mentioned earlier, when the app detects the user's

position near the underground entrance, it displays a prompt asking if the user has entered the tunnel. If the user confirms, the map switch is executed immediately, ensuring improved localization accuracy and a seamless user experience. If there is no response from the user, the app continues to monitor the altimeter data and triggers a map switch upon detecting significant changes. The reason for this design is that if the user does not respond, it may mean they have not entered the entrance, in which case the altimeter data remains relatively stable, and no action is needed. Alternatively, if the user does enter the entrance without responding, it suggests they may not require navigation information at that moment and are relying on their own judgment. In this case, the app can provide a slowly map switch. Notably, both the overground and underground maps provide overlapping areas near the tunnel entrance, allowing the app to offer flexible and seamless instructions regardless of the user's choice.

9.4.3 Challenging Spots: Finding Entrance to a Ramp

The previously described mechanism can be applied in multiple scenarios where localization accuracy is not sufficient. For instance, consider the situation where a user encounters a challenging spot, such as a downward ramp with a narrow entrance, parallel to the walkway, as illustrated in Figure 9.11.

In such case, achieving the required accuracy with the localization system can be challenging. The system may struggle to distinguish whether the user is on the walkway or the ramp (and vice versa). As a result, the app's instructions based on the incorrect user position can lead to user confusion. To address this issue, we employ a similar "User as a Sensor" approach: when the user enters the area near the ramp, the app prompts the user to confirm

Figure 9.11: The ramp that users need to negotiate, with a highlighted yellow path indicating the route.

whether they have entered the ramp. Subsequently, the app provides instructions based on the user's response. Once again, if there is no response from the user, we assume that the user is aware of their actions and offer default instructions that can be ignored by the user.

9.5 Experimental Results

In this section, we'll start by explaining the experiment's procedure and the user study details. Afterward, we'll examine the paths reconstructed during the experiment, focusing on both successful and unsuccessful instances of the particle filter-based localization method, along with related discussions.

9.5.1 Experimental Procedure

Here is the procedure for conducting experiments with this app.

1. The user must select a route from the list displayed on the app screen. These routes are predefined by the user through a front-end, which is part of a system developed by other contributors of this project.

2. The user is positioned at a known starting point and is required to walk in a specified direction for a certain distance (typically, 6 steps) to initiate the initial calibration process.

3. The navigation process then begins. Users can interact with the app using a Bluetooth bone-conduction earphone and VoiceOver. Although it is allowed for users to place the phone in their pockets, all users decided to hold the phone in their hands. The Route Compass feature assists users in determining the direction to their next target, and the app prompts users to respond when it is uncertain about their precise position.

4. The entire process concludes when the user reaches the destination of the current route.

9.5.2 User Study

In the user study, seven blind participants took part, with ages ranging from 59 to 74 and a mean age of 67. Among the participants, there were two females and five males. Three of the participants used guide dogs, while the remaining four relied on walking canes. The Table 9.1 provides more details.

9.5.3 Experiment Details and Localization Results

The task assigned to these users was to traverse three selected trajectories within the Palo Alto train station. We defined three starting/end points in the transit hub: Northbound,

ID	Age	Gender	Onset of blindness	Independent traveler?	Mobility aid
P1	74	M	Since teen age	Rarely travels alone	Long cane
P2	74	M	Since teen age	Yes (when younger)	Long cane
P3	60	M	Since birth	Yes	Long cane
P4	65	F	Since birth	Yes	Dog guide
P5	69	M	Since 3 years of age	Yes	Dog guide
P6	59	F	Since teen age	Yes	Dog guide
P7	70	M	Since birth	Yes	Long cane

Table 9.1: Participants' characteristics.

Southbound, and Bus stop. The trajectories are as follows. The visualization of these three trajectories can be found in the different figures this chapter, which are represented by white lines.

1. **NB2SB**: The first trajectory starts at Northbound and ends at Southbound.

2. **SB2BUS**: The second trajectory starts at Southbound and ends at the Bus stop.

3. **BUS2NB**: The third trajectory starts at the Bus stop and ends at Northbound.

It's worth noting that the first and third paths require users to navigate within an underground tunnel, making the trajectories more challenging and representative of real-life conditions. It's important to acknowledge that the study environment contains various challenges. Despite conducting the user studies on weekends, the station remained busy, with numerous

people standing, walking, and biking, and several unhoused individuals sitting or lying on the ground. The presence of periodically incoming diesel-powered trains contributed to the noise in the environment. Additionally, when users walked through the underground passageway, the intense car traffic in the nearby highway underpass added to the noise levels.

The traversal times for all participants on these three routes are displayed below, along with the length of each trajectory. Since outdoor navigation trajectories often take longer compared to indoor cases, we use minutes as the unit of measurement here, rather than seconds. It's worth noting that the traverse time for the "Bus stop to Northbound" trajectory for participant P6 is not available due to a localization system failure, which will be explained in the next section.

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**Length**
NB2SB	19	20	9	9	4	8	11	130 m
SB2Bus	27	20	13	22	11	21	10	145 m
Bus2NB	47	28	13	13	6	-	10	225 m

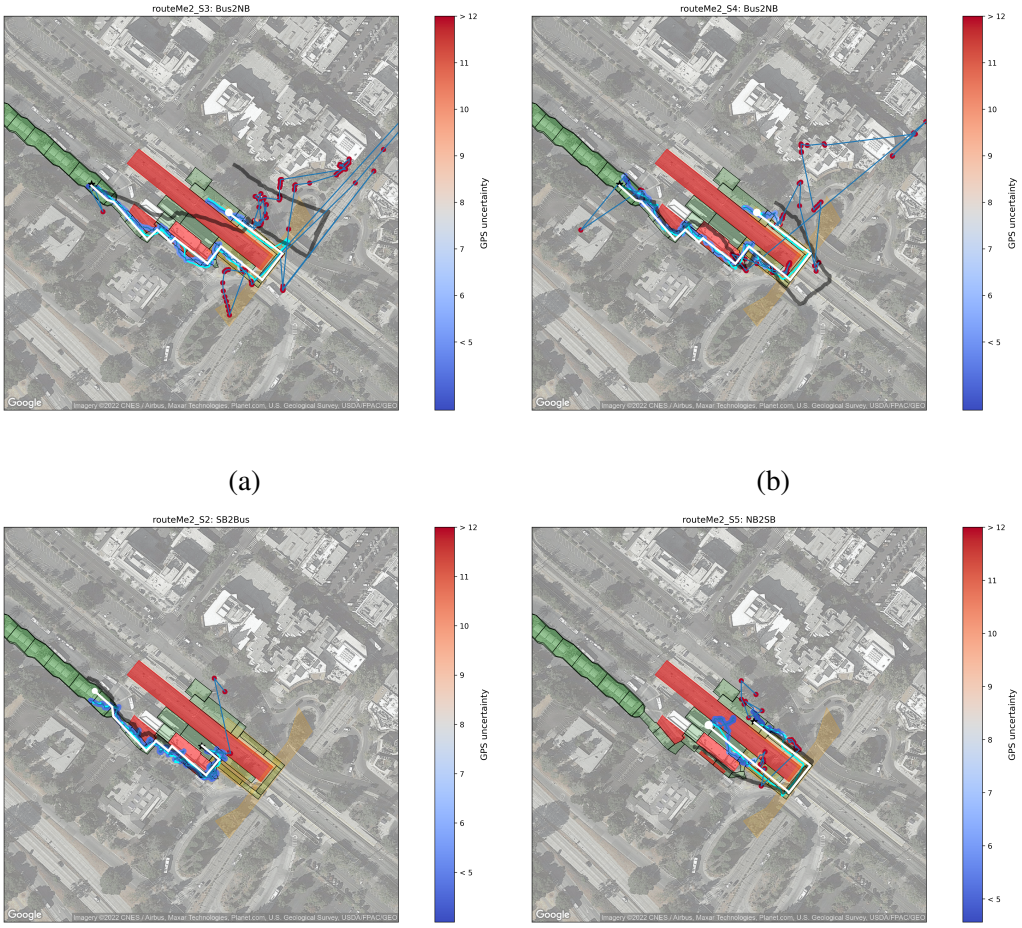
Table 9.2: Traversal times (in minutes) for all participants and all routes.

This app demonstrates the capability to accurately track the user's position in most instances, despite the presence of GPS errors and IMU sensor drift. It is noteworthy that the drift observed in the RoNIN trajectory can be rectified through the incorporation of GPS and map information (see Figure 9.12 (a)(b)). Furthermore, it is evident that the combination of GPS signal and map can correct the incorrect scalar of the RoNIN trajectory (see Figure 9.12

(a)(b)(d)). In Figure 9.12 (a)(b)(d), the particle filter continues to function when the user traverses the underground tunnel, even in areas where GPS signals are denied. Additionally, in Figure 9.12 (d), as the user reaches the Southboarding area, the GPS signal becomes significantly inaccurate, with the user's position erroneously placed in an No-go zone (the railway) with low accuracy. However, by leveraging the map information, the model manages to provide a correct trajectory. More visualization results can be found in Figure 9.14.

9.5.4 Challenges and Issues: Discussion

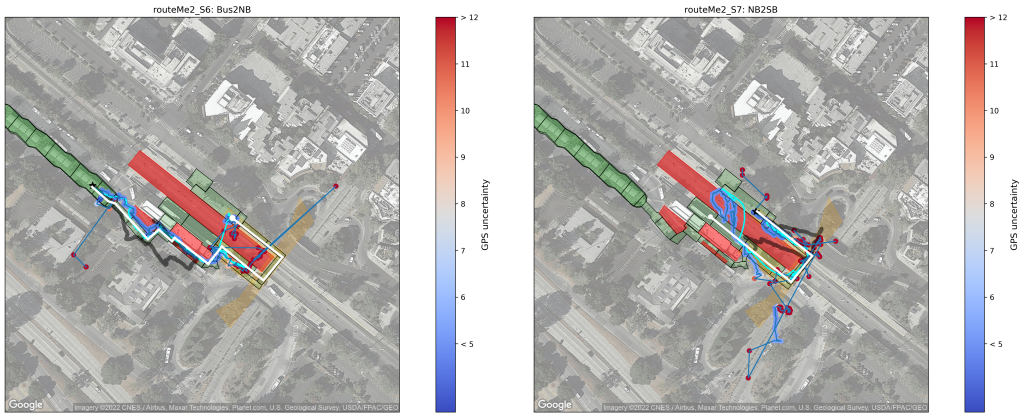
Although the app performs correctly in most cases, and all participants successfully complete their routes, there are two instances of failure due to incorrect user localization/tracking. For participant P7, at the end of the NB2SB route, the user's estimated location is inaccurately placed at the Northbound platform when it should be at the Southbound platform (Figure Figure 9.13 (b)). This mislocalization occurred because, when the user exited the tunnel, a significant and sustained GPS error with a very low uncertainty radius was observed. This erroneous GPS data led the system to erroneously trust the GPS accuracy, pushing all particles to the wrong side of the tracks. Another failure case involves participant P6 in the Bus2NB route (Figure 9.13 (a)). In this case, RoNIN undershot the user's position, making it impossible for the model to correctly track the user. Note that when the user entered the Northbound area, the fail-safe mechanism was activated, and the user's location was accurately corrected by the GPS signal.



(a) (b)

(c) (d)

Figure 9.12: Comparison of Localization Methods. The trajectories are represented as follows: Black line - RoNIN, Blue line - GPS (uncertainty visualized by color), Cyan line - Localization estimation results from the model. White line - Ground truth path, White Circle - Destination. (a) Participant 3, trail: Bus Stop to Northbound. (b) Participant 4, trail: Bus Stop to Northbound. (c) Participant 2, trail: Southbound to Bus Stop.(d) Participant 5, trail: Northbound to Southbound.



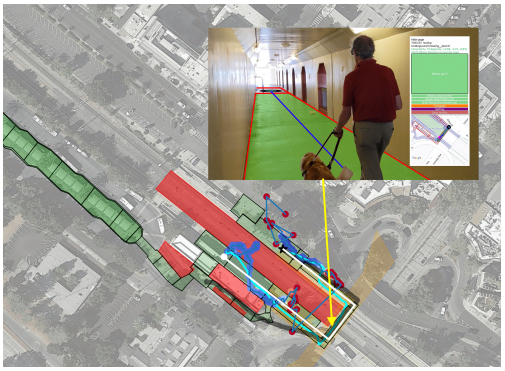
(a) (b)

Figure 9.13: Failure cases. Black line - RoNIN, Blue line - GPS (uncertainty visualized by color), Cyan line - Localization estimation results from the model. White line - Ground truth path, White Circle - Destination. (a) Participant S6, trail: Bus Stop to Northbound (b) Participant S7, trail: Northbound to Southbound

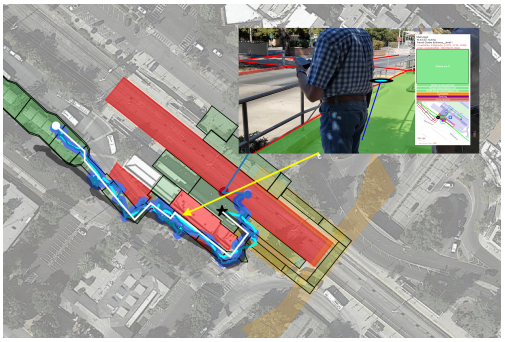
9.6 Conclusion

In this chapter, we have provided an in-depth exploration of the RouteNav navigation app, which assists users in navigating from their starting point to their chosen destinations. We began by discussing the limitations of using only IMU sensors or GPS for localization, highlighting the need for compensation with a map and sensor fusion. Then, we delved into the particle filter algorithm, detailing its parameters, and discussed the fail-safe mechanism implemented within the application.

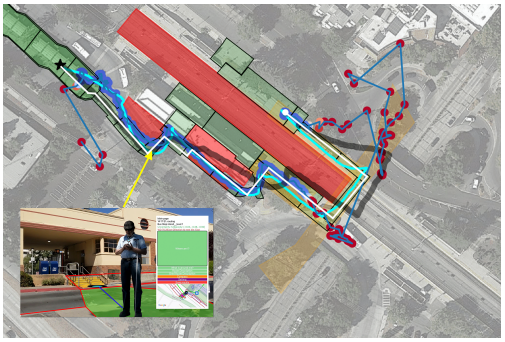
Subsequently, we focused on the specific user interface design and consider challenging navigation scenarios. We addressed the challenges associated with altimeter data and the difficulty in determining the optimal time to switch between overground and underground maps. We talked about the usage of ”user as a sensor” mechanism, which can help the user overcome



(a)



(b)



(c)

Figure 9.14: Experimental results visualization from three participants in different trials. The trajectories are represented as follows: Black line - RoNIN, Blue line - GPS (uncertainty visualized by color), Cyan line - Localization estimation results from the model. White line: Ground truth path, White Circle - Destination. Photos from participants are taken in the location pointed by the yellow arrow. (a) Participant 1, trail: from Northbound to Southbound. (b) Participant 3, trail: from Southbound to Bus stop. (c) Participant 5, trail: from Bus stop to Northbound.

sensor uncertainty effectively.

Moving forward, we talked about the experimental procedure and a user study, providing comprehensive details about the study participants, experiment design, and user trajectories. We concluded this section by presenting the results of our experiments, which included both successful and unsuccessful cases.

Chapter 10

Ablation Study

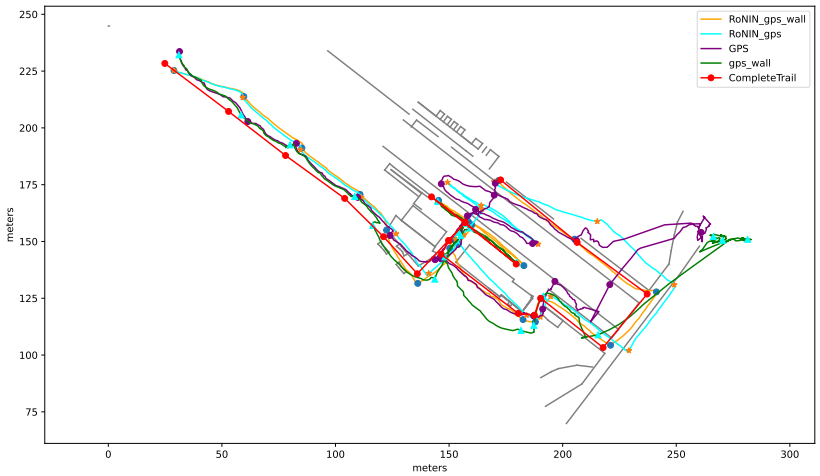
After discussing the experimental results involving visually impaired individuals in the transit-hub environment, an interesting question arises: how does the sensor fusion model perform when some of the information is omitted? In Chapter 9 we have already considered cases where RoNIN or GPS is utilized individually, and we have ascertained that this leads to poor results. However, what about other scenarios, such as RoNIN + GPS (without map information) or GPS + Wall (GPS data integrated with map information without relying on RoNIN)? Do these scenarios yield similar or even superior results compared to our standard model, RoNIN + GPS + Wall? We will explore these scenarios in this chapter. It's worth noting that another potential combination is RoNIN + Wall. However, given the numerous open spaces on the map, we have decided not to pursue this option since particles in open areas, without any constraints, are prone to disperse without restriction.

Here, we present the results collected during experiments conducted by myself and my advisor, Professor Manduchi, at the Palo Alto station during the summer of 2023. We used

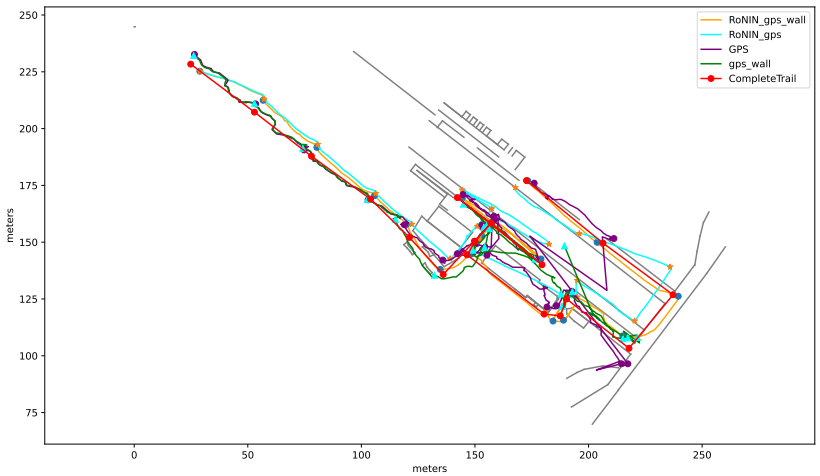
multiple iPhone, either held in our hands or placed in our pockets while walking. It's important to note that here we aimed to demonstrate that our system is effective and beneficial not only for blind individuals but also for sighted users. Six trajectories, start at index 1 are analyzed here. Ground truth path are shown in Figure 10.2. For trail 1,2,4,6, we start at the Bus stop and reach the NorthBound, while for trail 3,5, the whole trajectory is in a reversed order.

10.1 Observations

We have selected two trajectories from our collected data for a qualitative study among different modalities. As depicted in Figure 10.1, the GPS data can be highly inaccurate. In Figure 10.1 (a), the GPS readings placed the user within the railway area, significantly affecting the accuracy of the RoNIN + GPS results. GPS + Wall, on the other hand, yielded better results since the map prevented the particles from entering into the railway area. One may wonder how to integrate GPS information with map data when employing a particle filter, which relies on velocity for updating particle positions during the prediction step. To address this challenge, I adopted an approach in which I extracted the differences in GPS positions between consecutive readings and computed the corresponding velocities as inputs. However, it is worth noting that the combination of GPS and wall data proved to be insufficient when the GPS signal became unavailable within the underground tunnel (Figure 10.1(a)). Additionally, the velocity obtained from GPS, even when collected above ground, may not consistently reflect the user's actual velocity.



(a)



(b)

Figure 10.1: Results of the Ablation Study. Predictions made by each modality when the user is in the corresponding ground truth points are indicated using distinct symbols and colors: (1) RoNIN + GPS + Wall (orange line with blue circles), (2) RoNIN + GPS (cyan line with orange stars), (3) GPS (purple line with purple circles), (4) GPS + Wall (green line with cyan triangles), and (5) Ground truth (red line with red circles). (a) Trail 1. (b) Trail 5.

10.2 Quantitative Study Results: Complete Trail

Let's now delve into the quantitative results. The ground truth points were pre-defined as a series of stationary points. During data collection, we stood still at each ground truth point for several seconds. Later, when processing the data, we used the RoNIN magnitude to detect the timestamps of these stationary points. This allowed us to determine the predicted positions of the different modalities mentioned earlier when our actual position aligned with the ground truth points. As a result, we obtained two sets of points: ground truth points and predictions. The mean L2 error was used to measure the discrepancy between these points. The ground truth trajectory is displayed in Figure 10.2.

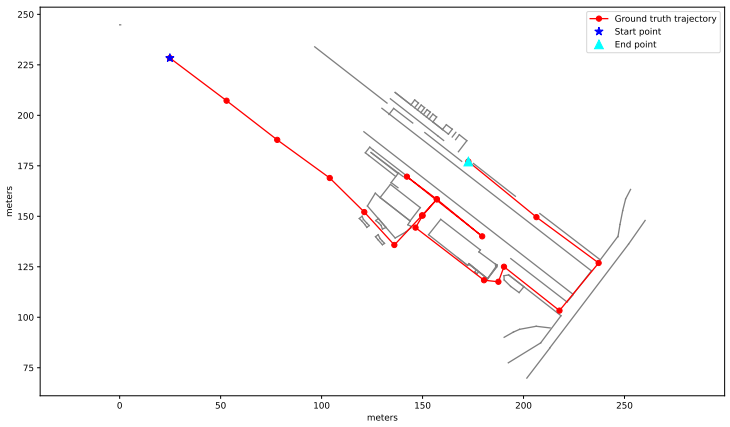


Figure 10.2: Ground Truth Trajectory, Ground truth points are shown in red circles: In trials 1, 2, 4, and 6, the user initiates their journey from the blue star point and aims to reach the destination marked using the cyan rectangle. Conversely, in trials 3 and 5, the trajectory is in reverse order.

Table 10.1 shows the comparison results of various modalities on the complete trajectories of Palo Alto train station. As observed, the combination of RoNIN, GPS, and Wall data

exhibits the lowest error, demonstrating its effectiveness. Combining RoNIN with GPS does not necessarily result in improved outcomes compared to using GPS alone, and the combination of GPS and wall data appears to worsen the outcomes. A more detailed understanding of these results will become clear as we analyze the performance of these modalities separately in various areas, a task we will address in the subsequent sections.

Trail Index	**RoNIN_GPS_Wall**	**RoNIN_GPS**	**GPS**	**GPS_Wall**
Trail 1	3.83	7.83	9.88	14.52
Trail 2	3.94	5.4	5.82	11.36
Trail 3	3.21	6.35	7.6	6.65
Trail 4	3.07	7.19	6.18	9.13
Trail 5	3.29	7.29	6.6	6.56
Trail 6	5.14	6.51	8.99	14.57
Mean	**3.75**	**6.76**	**7.51**	**10.47**

Table 10.1: Comparison results of the different modalities on the complete trail of Palo Alto train station (in meters).

10.3 Quantitative Study Results: Distinct Areas

One may also be interested in comparing these modalities in various areas. For instance, the comparison of open space area, the comparison of overground map areas, and the comparison of underground area. The ground truth points' definitions in different areas are represented using different colors in Figure 10.3. Considering these distinct areas, let us proceed

with separate comparisons.

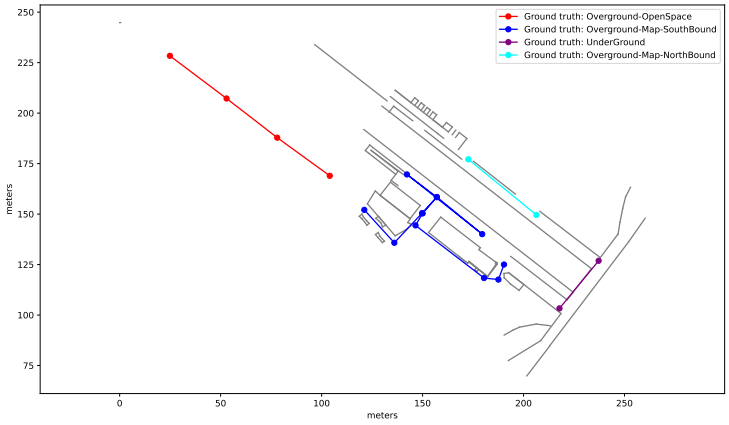


Figure 10.3: Ground truth points in different areas. Red points: Open space. Blue points: Southbound area. Purple points: Underground. Cyan points: Northbound area.

10.3.1 Open Space

For the ground truth points located in open spaces, it is evident that the particle filter has a minimal impact on the results due to the absence of map information. Therefore, the differences between RoNIN + GPS + Wall and RoNIN + GPS, GPS + Wall and GPS alone, are negligible, as one can see in Table 10.2. Additionally, the use of GPS data results in slightly lower errors compared to the combination of RoNIN and GPS, although with a relatively small difference. Notably, these disparities stem from the drift error within RoNIN and the random noise introduced at various stages of the particle filtering process (Figure 10.4(a)).

Trail Index	RoNIN_GPS_Wall	RoNIN_GPS	GPS	GPS_Wall
Trail 1	7.34	7.05	7.71	5.65
Trail 2	4.08	4.55	3.41	3.16
Trail 3	6.87	6.91	5.04	5.62
Trail 4	4.61	5.38	3.42	3.59
Trail 5	6.62	6.62	5.93	6.42
Trail 6	8.28	8.45	8.85	9.82
Mean	**6.3**	**6.49**	**5.73**	**5.71**

Table 10.2: Comparison results of the different modalities on the overground of Palo Alto train station in open space (in meters).

10.3.2 Overground Areas with Map information

Once the map information becomes available, we analyze the ground truth points separately for the Southbound area and the Northbound area. This separation is necessary due to the fact that Trail 3 and Trail 5 start from the Northbound, while other trails (Trail 1, 2, 4, 6) start from the bus stop. For Trails 1, 2, 4, and 6, GPS + Wall may face challenges when navigating the underground tunnel due to the potential unreliability of velocity extracted from the GPS trajectory in such conditions. As a result, the particles in this scenario may never reach the Northbound area, causing significant errors in the Northbound platform. Therefore, it is more informative to present the results in two separate tables for these two sets of points. Additionally, it's worth noting that for Trail 3 and 5, even when users start in the Northbound area and need to enter an underground tunnel, the GPS + Wall trajectory manages to reach the

other side of the railway. Therefore, it makes sense to include predictions from Trail 3 and 5 in the GPS + Wall modality in Table 10.3.

10.3.3 Overground: Southbound Area

Now, let's focus on the points near the Southbound area. As we can see in Table 10.3, RoNIN + GPS + Wall demonstrates superior performance compared to all other modalities. Comparing it with the results of RoNIN + GPS suggests the importance of map information. Additionally, when compared with GPS + Wall, one can conclude that RoNIN is another crucial contributor to the model's performance. Notably, combining RoNIN with GPS does not lead to improved results compared to using GPS alone, whereas the integration of GPS with wall information leads to enhanced performance, as observed in cases such as Figure 10.4 (b). This also underscores the significant impact of map information in improving outcomes.

Trail Index	**RoNIN_GPS_Wall**	**RoNIN_GPS**	**GPS**	**GPS_Wall**
Trail 1	3.07	7.39	8.28	4.2
Trail 2	3.53	4.1	3.45	4.22
Trail 3	1.99	5.15	6.66	4.52
Trail 4	2.61	6.72	4.77	5.21
Trail 5	2.22	6.56	6.52	4.76
Trail 6	5.23	6.06	5.93	6.42
Mean	**3.11**	**6.0**	**5.94**	**4.89**

Table 10.3: Comparison results for different modalities on the Palo Alto train station overground with maps, limited to data within the Overground-Map-SouthBound area (in meters).

10.3.4 Overground: Northbound Area

Results of the Northbound area are shown in Table 10.4. Within the table, we still observe that RoNIN + GPS + Wall remains the most effective option, while RoNIN + GPS performs worse than using GPS alone. For trails that commence at the bus stop (Trails 1, 2, 4, and 6), these discrepancies arise due to the absence of constraints provided by the map. In the case of Trails 3 and 5, where the user initiates their journey in the Northbound area without entering the underground tunnel, the drift from RoNIN can impact the final outputs. Regarding GPS + Wall, as explained in subsection 10.3.2, Trails 1, 2, 4, and 6 exhibit larger errors when compared to using GPS alone (Figure 10.4 (c)). For Trails 3 and 5, the results from GPS + Wall remain reasonable.

Trail Index	**RoNIN_GPS_Wall**	**RoNIN_GPS**	**GPS**	**GPS_Wall**
Trail 1	1.4	7.87	1.81	80.47
Trail 2	6.2	8.6	6.38	30.54
Trail 3	3.17	6.93	6.76	7.19
Trail 4	1.59	8.51	4.38	38.16
Trail 5	3.4	6.16	8.95	8.58
Trail 6	1.1	4.99	3.45	40.33
Mean	**2.81**	**7.18**	**5.29**	**34.21**

Table 10.4: Comparison results for different modalities on the Palo Alto train station overground with maps, limited to data within the Overground-Map-NorthBound area (in meters).

10.3.5 Underground

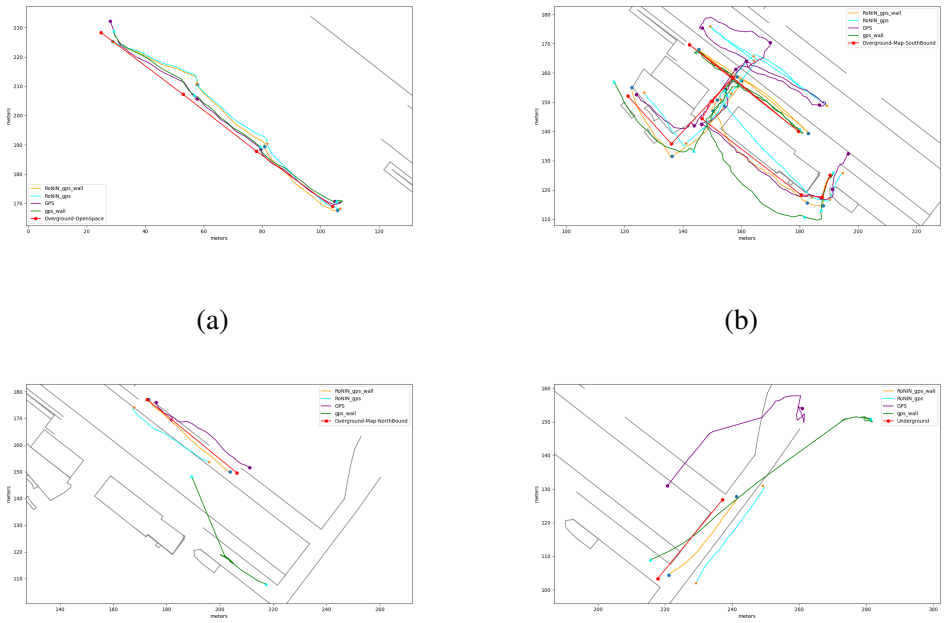
Now, let's examine the results within the underground tunnel (Table 10.5). As one might expect, the GPS signal is unusable here. While RoNIN clearly outperforms GPS, it still exhibits some drift, which can be alleviated through the inclusion of map information. Therefore, RoNIN + GPS + Wall produces the best results in this scenario (Figure 10.4(d)).

Trail Index	**RoNIN_GPS_Wall**	**RoNIN_GPS**	**GPS**	**GPS_Wall**
Trail 1	3.8	12.05	31.96	28.19
Trail 2	3.85	11.73	24.34	51.43
Trail 3	3.21	11.79	19.24	20.9
Trail 4	4.24	12.33	21.96	14.68
Trail 5	2.92	14.18	6.11	15.65
Trail 6	2.33	6.85	33.14	47.24
Mean	**3.39**	**11.49**	**22.79**	**29.68**

Table 10.5: Comparison results of the different modalities on the underground of Palo Alto train station (in meters).

10.4 Conclusion

In this chapter, we discuss the results of the ablation study conducted using six trajectories gathered in the Palo Alto transit-hub by two sighted individuals. We explore various combinations of information sources, including RoNIN + GPS, GPS + Wall, GPS alone, and RoNIN + GPS + Wall. Apart from visually comparing predictions to ground truth points, we



(a) (b)

(c) (d)

Figure 10.4: Results of the comparisons in various areas. Predictions made by each modality when the user is in the corresponding ground truth points are indicated using distinct symbols and colors: (1) RoNIN + GPS + Wall (orange line with blue circles), (2) RoNIN + GPS (cyan line with orange stars), (3) GPS (purple line with purple circles), (4) GPS + Wall (green line with cyan triangles), and (5) Ground truth (red line with red circles). (a) Open space, Trail 2. (b) Southbound area, Trail 1. (c) Northbound area, Trail 4. (d) Underground area, Trail 1.

conduct multiple quantitative analyses of the study results. Specifically, we initially compare the complete trail and subsequently delve into comparing results in distinct areas, including open space, the Southbound overground area, the Northbound overground area, and the underground section. These results together indicate that both map information and GPS data are essential components, with the combination of RoNIN + GPS + Wall producing the most favorable outcomes.

Chapter 11

Conclusion and Open Problems

11.1 Conclusion

This thesis focuses on the development of a smartphone-based indoor/outdoor localization system designed to assist visually impaired individuals. Various sensors used for localization have their limitations. BLE beacons and Wi-Fi require external infrastructure, and their fingerprinting-based method necessitates the creation and maintenance of a fingerprinting database, resulting in significant costs. Furthermore, determining the user's precise location during data collection can be challenging. The magnetic sensor, while not requiring external infrastructure, still faces the mentioned challenges when building its fingerprinting database. The UWB method, although not depending on a database, demands external infrastructure and specific hardware which is not universally available on smartphones. The camera sensor, while free from these drawbacks, demands a continuous, unobstructed line of sight, making it unsuitable for visually impaired travelers.

Our primary focus centers on IMU sensors. Despite inherent drift errors, IMU sensors can be employed for localization and tracking without the aforementioned limitations, making them particularly well-suited for visually impaired users. In scenarios without a map, we introduce a two-stage turn detection method based on the Mixture Kalman Filter (MKF) and Gated Recurrent Unit (GRU). Specifically, the MKF method utilizes azimuth angle data to ascertain the user's indoor orientation, subject to the constraint that the user's orientation is a multiple of either 45° or 90°. The MKF maintains a list of Kalman filters, each tracking a potential user orientation. At each timestamp, azimuth data is used as input for these filters to evaluate their alignment with the input. Depending on this alignment, each Kalman filter is assigned weights that determine their contributions to the final MKF output. The GRU makes use of user acceleration and azimuth data as inputs to determine if the user is walking in a straight line or not. The GRU outputs serve two valuable purposes: Firstly, we increase the turn probability of the MKF when the user isn't walking in a straight line. Furthermore, through the comparison of the user's facing direction before and after each non-straight walking interval, we can efficiently eliminate incorrectly identified turns, leading to improved outcomes.

When a map is available, the Particle Filter can be integrated with the map and Pedestrian Dead Reckoning (PDR) techniques to enable accurate localization. The velocity obtained from the PDR method serves as input for the particle filter's prediction step, while map data is employed to modify particle weights during the update step, followed by a resampling step based on these weights. The map is represented as a 2D matrix utilizing the Bresenham algorithm, optimizing the time-space trade-off to ensure efficient execution of the particle filter on a smartphone. Two PDR methods are examined: the first method, named Azimuth/Steps, using

Long Short-Term Memory (LSTM) for counting steps and a particle filter for estimating step length. The second method uses an open-source model called RoNIN, which is trained with data from sighted individuals and determines the user's walking direction instead of the phone orientation. Additionally, the Mean-Shift algorithm can detect multiple clusters of particles. Two methods for managing clustered data and providing model output are presented: the local maximum and global maximum algorithms. When GPS data is present, it is modeled through a bivariate Gaussian distribution with a flexible radius. This method enhances the precision of particle filter weight updates, particularly in open spaces and in cases where GPS positions may exhibit back-and-forth jumps.

In the experimental chapters, this thesis evaluates the two-stage turn detection algorithm and the particle filter algorithm using the WeAllWalk dataset, demonstrating their efficiency on data from visually impaired individuals. To examine these methods in real-world scenarios, two iOS applications are discussed: the indoor navigation app incorporates PDR methods with the particle filter, offering an intuitive interface seamlessly integrated with the Apple Watch. Furthermore, a user-friendly front-end is provided for managing essential map information. In the transit hub navigation app, the interface is designed to compensate localization uncertainty. Concepts like tiles and goalposts reduce the influence of positioning errors on notifications and directional instructions. A user-oriented route compass is introduced and favored by participants. Additionally, we implement a "User as a Sensor" mechanism, enabling users to collaborate with the application to address challenges posed by GPS outages and IMU drift errors, which may affect the model's accuracy. The efficiency of these two applications is confirmed through user studies, and our qualitative and quantitative ablation study results un-

derscore the effectiveness of integrating data from various sources, namely RoNIN, GPS, and maps.

11.2 Open Problems

Despite the significant progress achieved in this thesis, the research domain of utilizing smartphones to aid visually impaired individuals in navigation presents several unresolved issues. When employing the particle filter algorithm, the presence of multiple clusters necessitates further exploration to determine the optimal method for effectively conveying this information to guide users. For instance, in cases where two clusters are detected, a decision must be made regarding what information should be presented to the user through the interface. Should we provide the position of the larger cluster, both clusters, or simply inform the user of the uncertainty due to the presence of multiple clusters? The situation becomes more intricate when additional clusters are identified, each with varying distances from one another. Furthermore, while it may not be practical for visually impaired users to constantly hold their smartphones, occasionally taking the device out of their pocket to use the camera for obtaining an accurate absolute position could significantly improve the accuracy of the relative position provided by the particle filter. Another intriguing area of research would be to develop a model that accurately predicts the walking direction of visually impaired individuals, rather than just determining the orientation of their phone. Current systems like RoNIN, although capable of identifying walking direction, often yield inaccurate results when applied to data from visually impaired users. This issue persists even with fine-tuning. Addressing this challenge can be rather difficult due

to the limited availability of data from blind individuals, but it certainly worth exploration given its potential impact.

For the indoor localization app, automating the process of adding new buildings to the application is desirable. This would involve providing a floor plan image and having it automatically incorporated into the app. Subsequently, examining the scalability of the app by evaluating its performance across various buildings would be interesting. In case of the transit hub localization app, doing more experiments and explore the efficiency of different components of the localization model for sighted users who also navigate complex indoor/outdoor environments warrants investigation. Furthermore, for mobile platforms like Android that provide raw GPS data, investigating tight sensor fusion between GPS and the particle filter represents a promising direction for addressing outdoor pedestrian localization challenges. Given the rapid advancement of hardware, an intriguing topic emerges: can the particle filter be executed on a smartwatch instead of a smartphone? This could prove particularly valuable in the smartwatch industry, as it would ensure stability and trajectory reliability during GPS outages, especially in urban environments where GPS signals are often unreliable.

Concerning visually impaired individuals, there remain open challenges in designing the user interface. Adapting the interface to accommodate varying hardware capabilities and positioning accuracy is a interesting area of exploration. While this thesis has already discussed an interface designed for the Apple Watch and iPhone, leveraging the capabilities of the latest Apple Watch to assist blind individuals is another interesting possibility. This is especially applicable since the newest Apple Watch supports AssistiveTouch[1]. Moreover, it is interesting

[1] https://developer.apple.com/documentation/uikit/uiguidedaccessaccessibilityfeature/3089190-assistivetouch

to investigate how ChatGPT[2] can contribute to further enhancing the user interface. For instance, instead of presenting multiple buttons on the screen, a natural language interface could be implemented. In this approach, users could interact directly with the application through a Bluetooth earphone. Spoken input would be converted to text using the iPhone's Speech framework[3], combined with contextual map information, and presented to ChatGPT through specific prompts. ChatGPT's responses would then be synthesized into speech using the Speech Synthesis framework[4]. Implementing such a system could significantly improve the user experience for blind individuals, enabling them to focus solely on the information they require without the burden of processing unnecessary details. For example, a user might ask, "What is the most notable landmark nearby?" Rather than offering a list of landmarks for the user to review, ChatGPT could identify a significant landmark based on its understanding of the term "notable" and provide this landmark information directly to the user. To sum up, researchers in the field of navigation assistance for visually impaired individuals should consider exploring various new techniques to consistently enhance their solutions.

[2] https://chat.openai.com/
[3] https://developer.apple.com/documentation/speech/
[4] https://developer.apple.com/documentation/avfoundation/speech_synthesis/

Bibliography

[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020.

[2] Dragan Ahmetovic, Cole Gleason, Chengxiong Ruan, Kris Kitani, Hironobu Takagi, and Chieko Asakawa. Navcog: a navigational cognitive assistant for the blind. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 90–99, 2016.

[3] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A Al-Ammar, and Hend S Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5):707, 2016.

[4] Moustafa Alzantot and Moustafa Youssef. Uptime: Ubiquitous pedestrian tracking using mobile phones. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 3204–3209. IEEE, 2012.

[5] Ilias Apostolopoulos, Navid Fallah, Eelke Folmer, and Kostas E Bekris. Integrated online

localization and navigation for people with visual impairments using smart phones. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(4):1–28, 2014.

[6] Safar M Asaad and Halgurd S Maghdid. A comprehensive review of indoor/outdoor localization solutions in iot era: Research challenges and future perspectives. *Computer Networks*, 212:109041, 2022.

[7] Shiri Azenkot, Richard E Ladner, and Jacob O Wobbrock. Smartphone haptic feedback for nonvisual wayfinding. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, pages 281–282, 2011.

[8] Billie L Bentzen, Janet M Barlow, and Lee S Tabor. *Detectable warnings: Synthesis of US and international practice*. US Access board Washington, DC, 2000.

[9] Joydeep Biswas and Manuela Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *2010 IEEE international conference on robotics and automation*, pages 4379–4384. IEEE, 2010.

[10] Johann Borenstein, Lauro Ojeda, and Surat Kwanmuang. Heuristic reduction of gyro drift in imu-based personnel tracking systems. In *Optics and Photonics in Global Homeland Security V and Biometric Technology for Human Identification VI*, volume 7306, pages 244–254. SPIE, 2009.

[11] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, 20(2):100–106, 1977.

[12] Chris Calori and David Vanden-Eynden. *Signage and wayfinding design: a complete guide to creating environmental graphic design systems*. Wiley Online Library, 2015.

[13] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.1, 2018.

[14] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, Andrew Markham, and Niki Trigoni. Oxiod: The dataset for deep inertial odometry. *arXiv preprint arXiv:1809.07491*, 2018.

[15] Rong Chen and Jun S Liu. Mixture kalman filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(3):493–508, 2000.

[16] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[17] Yun Cheng and Taoyun Zhou. Uwb indoor positioning algorithm based on tdoa technology. In *2019 10th international conference on information technology in medicine and education (ITME)*, pages 777–782. IEEE, 2019.

[18] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[19] Adele Crudden, Jennifer L Cmar, and Michele C McDonnall. Stress associated with trans-

portation: A survey of persons with visual impairments. *Journal of Visual Impairment & Blindness*, 111(3):219–230, 2017.

[20] Ping-Jung Duh, Yu-Cheng Sung, Liang-Yu Fan Chiang, Yung-Ju Chang, and Kuan-Wen Chen. V-eye: A vision-based navigation system for the visually impaired. *IEEE Transactions on Multimedia*, 23:1567–1580, 2020.

[21] Navid Fallah, Ilias Apostolopoulos, Kostas Bekris, and Eelke Folmer. The user as a sensor: navigating users with visual impairments in indoor spaces using tactile landmarks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 425–432, 2012.

[22] Carl Fischer, Poorna Talkad Sukumar, and Mike Hazas. Tutorial: Implementing a pedestrian tracker using inertial sensors. *IEEE pervasive computing*, 12(2):17–27, 2012.

[23] German Flores and Roberto Manduchi. Easy return: an app for indoor backtracking assistance. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[24] German H Flores and Roberto Manduchi. Weallwalk: An annotated dataset of inertial sensor time series from blind walkers. *ACM Transactions on Accessible Computing (TACCESS)*, 11(1):1–28, 2018.

[25] Michael D Fowler. Soundscape as a design strategy for landscape architectural praxis. *Design studies*, 34(1):111–128, 2013.

[26] Giovanni Fusco and James M Coughlan. Indoor localization using computer vision and

visual-inertial odometry. In *Computers Helping People with Special Needs: 16th International Conference, ICCHP 2018, Linz, Austria, July 11-13, 2018, Proceedings, Part II 16*, pages 86–93. Springer, 2018.

[27] Giovanni Fusco and James M Coughlan. Indoor localization for visually impaired travelers using computer vision on a smartphone. In *Proceedings of the 17th international web for all conference*, pages 1–11, 2020.

[28] Reginald G Golledge, Roberta L Klatzky, Jack M Loomis, Jon Speigle, and Jerome Tietz. A geographical information system for a gps based personal guidance system. *International Journal of Geographical Information Science*, 12(7):727–749, 1998.

[29] Reginald G Golledge, Jack M Loomis, Roberta L Klatzky, Andreas Flury, and Xiao Li Yang. Designing a personal guidance system to aid navigation without sight: Progress on the gis component. *International Journal of Geographical Information System*, 5(4):373–395, 1991.

[30] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40:33–51, 1975.

[31] Brandon Gozick, Kalyan Pathapati Subbu, Ram Dantu, and Tomyo Maeshiro. Magnetic maps for indoor navigation. *IEEE Transactions on Instrumentation and Measurement*, 60(12):3883–3891, 2011.

[32] Fuqiang Gu, Kourosh Khoshelham, Chunyang Yu, and Jianga Shang. Accurate step length estimation for pedestrian dead reckoning localization using stacked autoencoders. *IEEE Transactions on Instrumentation and Measurement*, 68(8):2705–2713, 2018.

[33] Janne Haverinen and Anssi Kemppainen. Global indoor self-localization based on the ambient magnetic field. *Robotics and Autonomous Systems*, 57(10):1028–1035, 2009.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[35] Michael Horvat, Christopher Ray, Vincent K Ramsey, Tanya Miszko, Roger Keeney, and Bruce B Blasch. Compensatory analysis and strategies for balance in individuals with visual impairments. *Journal of Visual Impairment & Blindness*, 97(11):695–703, 2003.

[36] Malek Karaim, Mohamed Elsheikh, Aboelmagd Noureldin, and RB Rustamov. Gnss error sources. *Multifunctional Operation and Application of GPS*, pages 69–85, 2018.

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] Silviya Korpilo, Tarmo Virtanen, and Susanna Lehvävirta. Smartphone gps tracking—inexpensive and efficient data collection on recreational movement. *Landscape and Urban Planning*, 157:608–617, 2017.

[39] Heli Koskimäki. Avoiding bias in classification accuracy-a case study for activity recognition. In *2015 IEEE symposium series on computational intelligence*, pages 301–306. IEEE, 2015.

[40] Jan Kouba, François Lahaye, and Pierre Tétreault. Precise point positioning. *Springer handbook of global navigation satellite systems*, pages 723–751, 2017.

[41] Kai Kunze, Paul Lukowicz, Kurt Partridge, and Bo Begole. Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. In *2009 international symposium on wearable computers*, pages 149–150. IEEE, 2009.

[42] Masaki Kuribayashi, Tatsuya Ishihara, Daisuke Sato, Jayakorn Vongkulbhisal, Karnik Ram, Seita Kayukawa, Hironobu Takagi, Shigeo Morishima, and Chieko Asakawa. Pathfinder: Designing a map-less navigation system for blind people in unfamiliar buildings. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2023.

[43] Richard B Langley. Rtk gps. *Gps World*, 9(9):70–76, 1998.

[44] Tung-Sing Leung and Gerard Medioni. Visual navigation aid for the blind in dynamic environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 565–572, 2014.

[45] Jiangyan Lu, Kin Wai Michael Siu, and Ping Xu. A comparative study of tactile paving design standards in different countries. In *2008 9th International Conference on Computer-Aided Industrial Design and Conceptual Design*, pages 753–758. IEEE, 2008.

[46] Tobias Margiani, Silvano Cortesi, Milena Keller, Christian Vogt, Tommaso Polonelli, and Michele Magno. Angle of arrival and centimeter distance estimation on a smart uwb sensor node. *IEEE Transactions on Instrumentation and Measurement*, 2023.

[47] Salvador Martínez-Cruz, Luis A Morales-Hernández, Gerardo I Pérez-Soto, Juan P Benitez-Rangel, and Karla A Camarillo-Gómez. An outdoor navigation assistance system

for visually impaired people in public transportation. *IEEE Access*, 9:130767–130777, 2021.

[48] Fatemeh Mirzaei, Jonathan Lam, and Roberto Manduchi. Accurate self-localization in transit stations: A case study. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2020.

[49] GJ Morgan-Owen and GT Johnston. Differential gps positioning. *Electronics & Communication Engineering Journal*, 7(1):11–21, 1995.

[50] John Morris and James Mueller. Blind and deaf consumer preferences for android and ios smartphones. In *Inclusive designing: Joining usability, accessibility, and inclusion*, pages 69–79. Springer, 2014.

[51] Monica Navarro, Simon Prior, and Montse Najar. Low complexity frequency domain toa estimation for ir-uwb communications. In *IEEE Vehicular Technology Conference*, pages 1–5. IEEE, 2006.

[52] Jari Nurmi, Elena Simona Lohan, Stephan Sand, Heikki Hurskainen, et al. *GALILEO positioning technology*, volume 176. Springer, 2015.

[53] Emin Orhan. Particle filtering. *Center for Neural Science, University of Rochester, Rochester, NY*, 8(11), 2012.

[54] Alwin Poulose and Dong Seog Han. Uwb indoor localization using deep learning lstm networks. *Applied Sciences*, 10(18):6290, 2020.

[55] Jiuchao Qian, Jiabin Ma, Rendong Ying, Peilin Liu, and Ling Pei. An improved indoor localization method using smartphone inertial sensors. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–7. IEEE, 2013.

[56] Peng Ren, Jonathan Lam, Roberto Manduchi, and Fatemeh Mirzaei. Experiments with routenav, a wayfinding app for blind travelers in a transit hub. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–15, 2023.

[57] Timothy H Riehle, Shane M Anderson, Patrick A Lichter, Nicholas A Giudice, Suneel I Sheikh, Robert J Knuesel, Daniel T Kollmann, and Daniel S Hedin. Indoor magnetic navigation for the blind. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1972–1975. IEEE, 2012.

[58] Timothy H Riehle, Shane M Anderson, Patrick A Lichter, William E Whalen, and Nicholas A Giudice. Indoor inertial waypoint navigation for the blind. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5187–5190. IEEE, 2013.

[59] Michailas Romanovas, Vadim Goridko, Ahmed Al-Jawad, Manuel Schwaab, Martin Traechtler, Lasse Klingbeil, and Yiannos Manoli. A study on indoor pedestrian localization algorithms with foot-mounted sensors. In *2012 international conference on indoor positioning and indoor navigation (IPIN)*, pages 1–10. IEEE, 2012.

[60] Udo Roßbach. *Positioning and navigation using the Russian satellite system GLONASS*. Univ. der Bundeswehr München, Studiengang Geodäsie und Geoinformationen, 2001.

[61] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[62] Sean L Seyler, Avishek Kumar, Michael F Thorpe, and Oliver Beckstein. Path similarity analysis: a method for quantifying macromolecular pathways. *PLoS computational biology*, 11(10):e1004568, 2015.

[63] Saleh Shadi, Saleh Hadi, Mohammad Amin Nazari, and Wolfram Hardt. Outdoor navigation for visually impaired based on deep learning. In *Proc. CEUR Workshop Proc*, volume 2514, pages 97–406, 2019.

[64] Ahamed Shafeeq and KS Hareesha. Dynamic clustering of data with modified k-means algorithm. In *Proceedings of the 2012 conference on information and computer networks*, pages 221–225, 2012.

[65] E. Smitshuijzen. *Signage Design Manual*. Lars Müller, 2007.

[66] Jérôme Soubielle, Inbar Fijalkow, Patrick Duvaut, and Alain Bibaut. Gps positioning in a multipath environment. *IEEE Transactions on Signal Processing*, 50(1):141–150, 2002.

[67] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.

[68] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[69] Viet Trinh and Roberto Manduchi. Semantic interior mapology: A toolbox for indoor scene description from architectural floor plans. In *24th International Conference on 3D Web Technology*, volume 2019. NIH Public Access, 2019.

[70] Raghav H Venkatnarayan and Muhammad Shahzad. Enhancing indoor inertial odometry with wifi. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1–27, 2019.

[71] Michal Vorlíček, Tom Stewart, Jasper Schipperijn, Jaroslav Burian, Lukáš Rubín, Jan Dygrỳn, Josef Mitáš, and Scott Duncan. Smart watch versus classic receivers: Static validity of three gps devices in different types of built environments. *Sensors*, 21(21):7232, 2021.

[72] 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, 2012.

[73] Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. Bluetooth positioning using rssi and triangulation methods. In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 837–842. IEEE, 2013.

[74] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[75] Oliver J Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory, 2007.

[76] Yuan Wu, Hai-Bing Zhu, Qing-Xiu Du, and Shu-Ming Tang. A survey of the research status of pedestrian dead reckoning systems based on inertial sensors. *International Journal of Automation and Computing*, 16:65–83, 2019.

[77] Shixiong Xia, Yi Liu, Guan Yuan, Mingjun Zhu, and Zhaohui Wang. Indoor fingerprint positioning based on wi-fi: An overview. *ISPRS international journal of geo-information*, 6(5):135, 2017.

[78] Guochang Xu and Yan Xu. *GPS*. Springer, 2007.

[79] Hang Yan, Sachini Herath, and Yasutaka Furukawa. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods. *arXiv preprint arXiv:1905.12853*, 2019.

[80] Hang Yan, Qi Shan, and Yasutaka Furukawa. Ridi: Robust imu double integration. In *Proceedings of the European conference on computer vision (ECCV)*, pages 621–636, 2018.

[81] Yuanxi Yang, Weiguang Gao, Shuren Guo, Yue Mao, and Yufei Yang. Introduction to beidou-3 navigation satellite system. *Navigation*, 66(1):7–18, 2019.

[82] Chris Yoon, Ryan Louie, Jeremy Ryan, MinhKhang Vu, Hyegi Bang, William Derksen, and Paul Ruvolo. Leveraging augmented reality to create apps for people with visual disabilities: A case study in indoor navigation. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*, pages 210–221, 2019.

[83] Yuan Zhuang, Jun Yang, You Li, Longning Qi, and Naser El-Sheimy. Smartphone-based indoor localization with bluetooth low energy beacons. *Sensors*, 16(5):596, 2016.

[84] Zheng Zuo, Liang Liu, Lei Zhang, and Yong Fang. Indoor positioning based on bluetooth low-energy beacons adopting graph optimization. *Sensors*, 18(11):3736, 2018.