UNIVERSITY OF CALIFORNIA
SANTA CRUZ

SAFERETURN: AN INTEGRATED INDOOR BACKTRACKING 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

Chia Hsuan Tsai

June 2024

The Dissertation of Chia Hsuan Tsai
is approved:

Dr. Roberto Manduchi, Chair

Dr. James Davis

Dr. Marcella Gomez

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

SafeReturn: An Integrated Indoor Backtracking System for Visually Impaired People

by

Chia Hsuan Tsai

Navigating indoors without a map can be challenging, especially for visually impaired individuals in unfamiliar settings. Many existing indoor navigation methods rely on building maps, pre-deployed infrastructure (like BLE beacons or RFID), or visual-based systems that require a clear line of sight to a camera. These requirements can make technology less accessible to visually impaired individuals for independent navigation.

To address these challenges, this thesis introduces the SafeReturn app, a new smartphone-based technology implemented as an iOS app designed to assist visually impaired individuals in returning to their starting point after navigating through indoor spaces. SafeReturn is particularly useful in real-life scenarios, such as hospitals, where a blind individual may initially navigate from a waiting room to a doctor's office with the assistance of a receptionist but subsequently needs to return independently. This technology eliminates the need for pre-deployed infrastructure or a clear camera view, enabling visually impaired individuals to navigate independently and confidently.

The system features a new path-matching algorithm enhanced by a hybrid matching approach that integrates magnetic field and inertial data (representing steps/turns information) to backtrack paths. When active, it records sensor data and provides guidance for users seeking to retrace their route. Additionally, it includes an

off-route detection mechanism that alerts users when they deviate from the correct path and provides notifications for path recovery to guide them back on track.

Initial testing was conducted using the WeAllWalk dataset containing inertial data from blind walkers. Subsequently, the system was equipped with a watch-based user interface and speech-based notifications specifically designed to simplify interaction for blind users. A user study involving seven visually impaired participants at the University of Santa Cruz's BE building demonstrated the effectiveness of the proposed localization solutions. The results from these tests illustrate the system's efficacy in assisting visually impaired individuals with indoor navigation and path recovery in real-world settings.

Acknowledgments

The completion of this thesis could not have been possible without the expertise and guidance of Dr. Roberto Manduchi. His mentorship has been transformative, offering not only invaluable academic insights but also giving me one of the most valuable second chances in my life: back to school to pursue my PhD degree. His belief in my potential, support, and patient guidance have shaped my research and personal growth. He always patiently listens to my research ideas during office hours and helps me refine them. I am deeply grateful for his mentorship, which has empowered me to navigate the complexities of academia with confidence and clarity.

I also want to express my gratitude to Dr. James Davis and Dr. Marcella Gomez for their pivotal roles as members of my defense committee. Since my qualification exam, they have offered invaluable insights, feedback, and encouragement to help me on my research and career path. Their mentorship has been instrumental in my growth as a researcher. I am also grateful to collaborate with Fatemeh Elyasi and Peng Ren for assistive technology. It's my pleasure working with you two, and this project has been so meaningful.

I want to thank my parents for believing in my dreams and supporting me throughout every challenge I faced. Their encouragement and resilience have inspired me to keep going, even when things get tough. Their sacrifices and guidance have shaped me into the person I am today, and eternally grateful.

A special deepest appreciation to my husband, Howard Yang. From the moment I resigned from my job to pursue my dream, he has been there every step of the way. His selflessness, encouragement, and willingness to shoulder additional responsibilities have provided me with the stability and motivation needed to overcome challenges and reach my goals. I am profoundly grateful for his presence, which has been a constant source of strength and inspiration throughout this journey. Words cannot fully express my gratitude. Thank you, Howard!

To my beloved children, Meya, Euna, and Kyron, thank you for your understanding and love. Over these past six years, Meya blossomed into a lovely teenager with a lot of cool ideas, Euna always tries to ease my worries and bring a smile to my face, and Kyron has grown from an infant into a talkative boy. As I promised you, Mommy will spend more time with you after graduation!

I also extend my appreciation to my brother and to all the individuals—teachers, friends, and colleagues—who have supported me along the way. Your encouragement, and friendship have been invaluable, and I am gratefully meeting/knowing/having each of you in my life.

Chapter 1

Introduction

Wayfinding in an unfamiliar environment could be challenging and potentially unsafe for visually impaired people because it is difficult to recognize landmarks at a distance or any other visual information. Path integration is one of the heavily used mechanisms for traversing a route for visually impaired people. While some visually impaired people can develop more precise spatial information about the route, others may build limited one-dimensional information about the route [1]. Systems that help visually impaired people with wayfinding could improve their opportunities for learning, employment, independent living, and social engagement. A widely used localization technology, GPS, is known for its high accuracy. However, it is relatively difficult to use it for indoor navigation because GPS signals are usually blocked indoors.

Several studies have explored methods to provide reliable indoor wayfinding for visually impaired individuals. Traditional techniques like BLE beacon and RFID deployment have been used for indoor positioning but require extensive infrastructure and environment fingerprinting, which can be time-consuming and costly [2][3][4]. Alternatively, smartphone-based systems leveraging the phone's built-in sensors for

pedestrian tracking in GPS-denied environments have gained popularity. Studies indicate that visually impaired individuals increasingly rely on smartphones in daily life [5].

The utilization of visual sensors, such as smartphone cameras, along with powerful AI systems to extract positional data and provide real-time information to users, including blind travelers, has gained significant interest in recent years. Smartphone-based approaches like visual-based odometry using Apple's ARKit are specifically designed to assist blind individuals [6][7][8]. While this method does not require infrastructure like beacons or RFID, it relies on a clear camera view. It may not always be feasible or convenient for blind travelers who typically use a long cane or guide dog, leaving one hand occupied. It has been commonly observed that navigational aids for blind individuals should ideally be hands-free [9][10]. Another hands-free option is Wi-Fi based navigation, leveraging existing Wi-Fi access points without additional infrastructure installation [11]. However, Wi-Fi based navigation requires a laborious fingerprinting operation to capture "signature" features such as the vector of received signal strength (RSSI) from Wi-Fi beacons. This process can be time-consuming and may pose challenges for widespread adoption in indoor navigation systems for visually impaired individuals.

Another smartphone-based approach is inertial odometry, specifically using Pedestrian Dead Reckoning (PDR), which has been studied for providing navigation assistance to visually impaired individuals [12][13][14][15]. Additionally, RoNIN is a learning-based algorithm that processes inertial data using neural network architectures to

generate motion vectors referenced to a fixed world frame [16]. One advantage of these techniques is that the phone's position on the body is not restricted. However, low-cost inertial sensors used in these systems may suffer from bias and noise issues, leading to directional and locational inaccuracies once integrated.

Alternatively, indoor navigation can utilize the magnetic field detected from the environment [17], which does not require specific infrastructure installation. The magnetic field in different indoor locations has unique signatures that can differentiate locations. Studies have explored using magnetic fields for indoor navigation for visually impaired individuals [18], some of which combine magnetic field data with inertial odometry systems to achieve higher accuracy [19][20].

Most applications mentioned earlier require access to indoor maps, which are not always publicly available. Even when building maps are accessible, certain methods like Wi-Fi based and magnetic field-based navigation require laborious fingerprinting operations to collect signal maps within the environment. This process can demand significant time and resources, particularly in large indoor spaces. As a result, we propose a solution that utilizes assisted return, a specific form of wayfinding, to provide hands-on indoor navigation for people with visual impairments without relying on pre-existing maps.

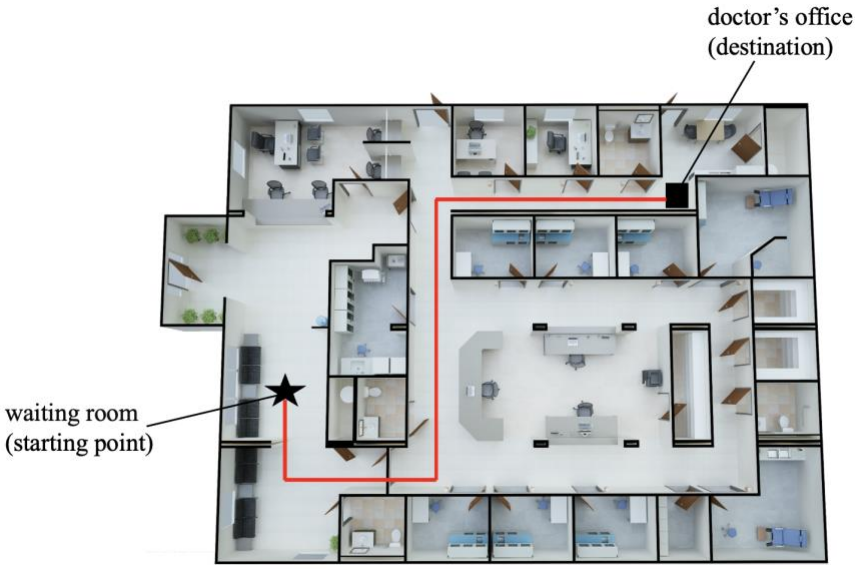


Figure 1.1: A hypothetical path of a blind patient for a doctor's appointment. The patient begins at the waiting room (marked by a star) and is guided by the receptionist to the doctor's office (marked by a square). After the appointment, the patient retraces the route back to the waiting room (from the square to the star).

The concept of assisted return was introduced by Flore and Manduchi [21], which assists visually impaired individuals in retracing their steps back to a starting point after navigating along a specific path. Figure 1.1 illustrates this concept with an example scenario: a blind patient begins at a waiting room for a doctor's appointment and is guided by a receptionist to the doctor's office. After the appointment, the same receptionist may assist the patient in returning to the waiting room. However, in some cases, the receptionist may not be available to help, or the blind patient may attempt to navigate back independently by remembering the turns and steps taken. An assisted return system is designed to help users retrace their path to return to the starting point. This thesis is inspired by Flores and Manduchi's Easy Return system, which relied solely on steps/turns information to aid visually impaired individuals in backtracking

their routes. However, that approach has its limitations. For instance, inaccuracies in step counting could lead to delayed or incorrect directions. In addition, the Easy Return system does not provide path recovery guidance when users deviated from the correct route. To address these shortcomings, we developed the SafeReturn app, which incorporates magnetic field information with a novel path-matching algorithm to enhance navigation accuracy and provide robust path recovery guidance.

Our assisted return system includes the following three tasks: (1) during the “way-in” path(walking from a starting point to a destination), tracking the user’s position; (2) during the “return” path, matching the current sub-path with the recorded way–in path (where the way-in path is time-reversed); (3) providing directions to the user during return with an appropriate user interface (including providing an overall description of the path). Since the system targets individuals with blindness, prioritizing datasets collected from visually impaired walkers over those from sighted individuals is critical. We adopted the WeAllWalk dataset [22], gathered previously by a prior PhD researcher in our group, consisting of annotated inertial data collected from ten blind individuals . Each participant carried two iPhones while they walked through several predefined trajectories using a walking cane or a dog guide. This dataset was explicitly used to test our system and was designed for visually impaired users. Our system was also tested in a user study involving seven visually impaired participants. During the study, participants traversed three predefined paths, each containing 4 to 5 turns, to assess the system's performance in real-world situations.

This thesis focuses on developing a smartphone-based “assisted return” system, SafeReturn, that offers real-time and reliable navigation assistance in mapless indoor environments for visually impaired individuals. We proposed a new path-matching algorithm that integrates the magnetic field and the inertial data (the input for our steps/turns detector developed by other PhD researchers in our group) to provide navigation. The system's performance was evaluated in a user study involving seven visually impaired participants. It is structured as follows:

In Chapter 2, we discuss indoor positioning systems, followed by a focused exploration of the "assisted return" system, a specific form of wayfinding.

Chapter 3 introduced a new path-matching algorithm considering walker orientation variations. We also discuss the challenges associated with backtracking a walker's position and highlight the necessity for a new matching method, further detailed in Chapter 6.

Chapter 4 discussed calibration techniques for magnetometers and explored how to leverage differences in magnetic fields between mapped locations (referred to as the "cost of magnetic field") in our path-matching algorithm.

In Chapter 5, we conducted simulations using the WeAllWalk dataset for assisted return, showing that using 90^o turn + steps information in our system provides significantly superior results. Additionally, we developed an iOS app to perform on-site tests, emphasizing the importance of the new hybrid matching algorithm.

Chapter 6 proposes a hybrid matching algorithm for determining the reliability of mapped positions by identifying the "last reliable position (LRP)." This chapter presents and compares two methodologies for identifying LRP.

In Chapter 7, we present the user study of the SafeReturn system involving seven visually impaired participants. This study evaluates the system's performance in real-life scenarios, covering aspects such as the app's user interface, experiment details, and results. The positive feedback from participants indicates increased feelings of safety and confidence while using the app, validating the potential of our proposed technology.

Finally, Chapter 8 provides a comprehensive conclusion to the study, summarizing key findings and discussing encountered challenges and proposed solutions.

Chapter 2

Related work

Indoor navigation systems have diverse applications across multiple scenarios. For instance, an indoor navigation system can guide hospital staff and visitors to specific departments within large medical buildings, ensuring efficient movement and timely access to critical areas. Moreover, indoor navigation enhances visitor experiences in museums, exhibitions, and conferences, enabling them to navigate complex layouts and explore exhibits effortlessly. These varied applications highlight the significance of indoor navigation systems in enhancing efficiency, safety, and user satisfaction across different environments. This chapter will explore various positioning methods used in indoor navigation systems.

2.1 Wi-Fi Based Indoor Positioning

Wi-Fi based indoor positioning employs two main approaches: signal strength-based and fingerprint-based positioning[23]. In signal strength-based positioning, the strength of the received signal is used to calculate the user's position using different methods (e.g., trilateration). On the other hand, fingerprint-based positioning requires the precollection of environment fingerprints in a database. The user's position is

determined by matching the received Wi-Fi signal with the pre-collected database. However, this method presents challenges, such as the time-consuming process of updating the fingerprint database and signal attenuation when passing through obstacles like walls or furniture.

2.2 BLE-Beacon Based Indoor Positioning

BLE-Beacon (Bluetooth Low Energy Beacon) is a low-cost alternative to traditional beacons, consuming less power. Similar to Wi-Fi based indoor positioning, BLE-Beacon mainly utilizes two positioning methods: strength-based and fingerprint-based. However, BLE-Beacon encounters similar challenges, with signal strength affected by indoor obstacles and difficulty maintaining the fingerprint database [24].

2.3 Inertial Sensor-Based Indoor Localization

With the widespread use of smartphones, inertial sensors (gyroscopes, accelerometers) have become integral to indoor positioning. One of the main advantages is that it does not require pre-deployed infrastructure and works even when smartphones are in users' pockets. However, inertial sensors are prone to bias and noise, leading to directional and locational inaccuracies [25][26]. Below are methods focused on enhancing its localization accuracy.

2.3.1 Strapdown Inertial Navigation

"Strapdown Inertial Navigation" refers to a specific method where inertial sensors are rigidly attached ("strapped down") to the moving object, such as a smartphone attached to a user. In strapdown inertial navigation, sensor measurements require further processing to provide corrections for generating the user's trajectory. The phone's orientation relative to an initial reference frame (with the Z axis pointing down) can be calculated by integrating the angular rate. User acceleration can then be determined by rotating accelerometer readings to the initial reference frame and subtracting gravity. Finally, position estimation is achieved through double integration of acceleration. Implementing a Kalman filter in this process can improve accuracy [26].

2.3.2 Pedestrian Dead Reckoning (PDR)

PDR is one of the simplest ways to track users' positions based on their steps and azimuth at each time step. Azimuth is obtained from post-processing and integrating sensor data from gyros and accelerometers [27]. However, PDR error accumulates over time due to sensor bias and noise. A solution to this shortcoming was a two-stage system consisting of a "straight-walking" detector and a Mixture Kalman Filter (MKF) for tracking orientation drift [12]. This system detects steps using an LSTM recurrent network [28]. In buildings with a structure represented by a network of corridors, a path can usually be described as a sequence of straight segments and turns with discrete turning angles (typically, multiples of 90º or 45º). It is called *turns/steps* representation. The robust turns/steps detector effectively reduces accumulated error in PDR[29].

2.4 Learning-Based Odometry

Neural networks have been widely used in learning-based odometry applications in recent years . Among these methods, RoNIN is one of the models that takes the inertial data to estimate the user's position and achieves a good performance [30]. One advantage of RoNIN is that it works regardless of smartphone position related to the body, producing motion vectors relative to a fixed world frame. However, RoNIN is susceptible to drift caused by inaccuracies in the accelerometer and gyroscope.

2.5 Magnetic Field Indoor Positioning

The utilization of indoor magnetic fields for positioning and navigation leverages the distinct characteristics of magnetic anomalies within indoor environments [19][31], [32][33]. Typically, these anomalies result from the combination of the Earth's magnetic field and the presence of ferromagnetic objects within the indoor space.

Several methodologies have been explored to leverage indoor magnetic fields for navigation purposes:

- **Fingerprint-based Positioning** [34][35][36]: This method involves pre-collecting a map of the magnetic field within the environment. Subsequently, a fingerprint-based positioning technique is utilized to determine the user's location within this mapped magnetic field, as shown in Figure 2.1. This

approach offers precise positioning but requires extensive offline data collection and processing.

- **Leader-Follower Model** [19][18][15]: In this model, a designated leader traverses the intended route, effectively mapping the magnetic field variations along the path. The follower then utilizes this mapped information to navigate through the environment. A similar concept was implemented by a prior PhD researcher in our lab, known as the Easy Return app. However, this app only utilized turns/steps data as input, without incorporating magnetic field data. This is also the model used in our application. While this approach reduces the need for extensive pre-mapping, it still requires a leader to traverse the route initially. Some studies have relied solely on the magnitude of the magnetic field for navigation [18][15]. However, research has shown that including directional information about the magnetic field (such as 2D or 3D magnetic field data) can improve navigation accuracy [37]. Magnetic field data can sometimes be integrated with other navigation methods to enhance location accuracy further. For instance, the FollowUs app [19] integrates inertial-based localization and magnetic field data for navigation. However, similar to [18] and [15], FollowUs relies on the magnitude of the magnetic field rather than its vectorized form. In addition, this app is designed for sighted users who can easily manage system errors and accuracies. For visually impaired people, building a system with a higher level of direction accuracy is necessary. In contrast, our system employs a novel path-matching algorithm that utilizes 2D magnetic field data and

incorporates layers of graphs to address orientation differences among mapped positions.

- **Landmark Information** [38]: Magnetic field measurements can be analyzed to extract landmark information; for example, the magnitude of the magnetic field may have peaked when the sensor is close to a pillar or another distinctive structure. During the map collection phase, one can record the landmark information and the corresponding magnetic field. During the navigation phase, the system can use the measured magnetic field to provide landmark information. This application is beneficial in environments with poor lighting conditions, allowing users to receive landmark information even without visual cues. This additional data enhances positioning accuracy, particularly when combined with other technologies like inertial-based localization.
- **Integration with Other Positioning Technologies** [39][19][40]: As mentioned earlier, magnetic field data can be combined with other positioning technologies, such as visual- or inertial-based localization, to improve overall accuracy. Since smartphones can easily gather both magnetic field and inertial data, this integration is feasible and enhances the reliability of indoor positioning systems.

However, most applications require an offline phase for data collection and processing. This process can be time-consuming and resource-intensive, particularly when mapping extensive indoor environments. For instance, the magnetic map in Figure 2.1

was created by dividing the mapping space into 2D grids with dimensions of 0.5m x 0.5m, and each grid contains magnetic measurements. In a typical indoor environment, such as a 20m x 20m area, there could be thousands of grids, each requiring the identification of its location and recording of magnetic field readings. This makes the process of building a magnetic map quite intensive [40].

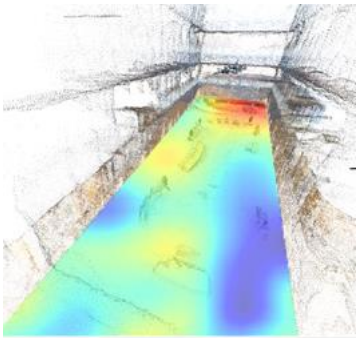


Figure 2.1: An example of a magnetic map [40]. The colored area indicates the different intensity of the magnetic field.

2.6 Assisted Return System

Assisted return is one of the particular forms of wayfinding that provides support to visually impaired people who, after walking along a certain path, are trying to trace their way back to the starting point. The concept is firstly introduced by Flore and Manduchi [21] and it is especially useful for navigation when the map of an indoor building is unavailable. The system could also help a person follow a path previously taken by another individual. Figure 2.2 shows an example of an indoor path that can be traced back by an assisted return system. As mentioned earlier, an assisted return

system includes three tasks: (1) during the “way–in” path tracking the user’s position;(2) during the “return” path, matching the current sub-path with the recorded way–in path; (3) providing directions to the user during return with an appropriate user interface.

Figure 2.2: An illustration of an indoor path typical for an assisted return system. This path comprises a series of straight segments connected by left or right turns [21].

Several technologies could be implemented for assisted return. Still, most of them require pre-deployed infrastructures in the environment (for example, preinstalled BLE-Beacon in the building for navigation to retrace the locations) or require prior knowledge of the map of the building. Here, we focus on a more hands-on technology that could be easily acquired in daily life to perform assisted return, smartphones. There have been studies that use smartphones to accomplish similar tasks. Those technologies can be divided into the following two categories.

2.6.1 Inertial Sensor-Based Assisted Return

Flore and Manduchi introduced the concept of assisted return by developing an inertial sensor-based app called Easy Return, which involved a study with six visually impaired participants navigating a controlled indoor environment. The Easy Return system utilizes inertial data to track steps and detect turns as users traverse a path within a building. When the user retraces their steps back to the starting point, the system compares their current position against the recorded path and provides directions based on remaining turns and steps. This system is also designed to handle scenarios such as mistaken turns or inaccuracies in step counting to facilitate route correction [21]. However, relying solely on inertial sensors can lead to positioning inaccuracies due to accumulated bias. To address this issue, we proposed an integrated system that combines magnetic field with steps/turns information.

Another system called FollowUs [19] operates on a peer-to-peer navigation model, where a leader (another user) who has previously traveled the route builds a map using inertial and magnetic field data for followers to replicate. However, FollowUs is primarily designed for sighted users who can manage system errors and accuracy effectively. A system tailored for visually impaired individuals requires a higher level of directional accuracy and a robust path recovery mechanism to ensure users can navigate back along the correct route.

2.6.2 Visual Odometry-Based Assisted Return

Clew [7] is an app based on visual odometry, specifically designed for visually impaired users to retrace their routes after exploring their surroundings. This app utilizes Apple's ARKit to estimate the user's movements in 3D space with high accuracy. However, a drawback of Clew is that it is not hands-free technology, which means users may need to hold both the smartphone with the app and another mobility aid like a long cane or guided dog simultaneously, potentially creating extra work for individuals with visual impairments.

Chapter 3

Path-Matching Algorithm

When a map of an indoor environment is not available, an assisted return system is designed for a visually impaired walker who has traversed a certain way-in route (possibly with the aid of a sighted companion) to traverse the same route in reverse (return), as shown in Figure 3.1. At any time instant during return, the system matches the current sub-path to a prior acquired (reversed) way-in path. The system's main task is matching any spatial information acquired during way-in with that acquired during return. In addition, the system offers directional guidance, including distance to the next turn and turn-by-turn instructions, and alerts users when they deviate from the intended route, facilitating path recovery.

If the odometry can be accurately recovered, one could simply match the current position estimated during return with the closest position in the way-in path. Unfortunately, large errors can be expected when relying only on inertial sensors. Hence, a more sophisticated strategy is necessary. Similarly to [18], we cast the problem as one of sub-sequence alignment, which seeks the best matching of the time sequence of measurements up to the current time during return, with an initial sequence of measurements during way-in. The measurements considered in this thesis include

information on steps/turns and variations in the magnetic field across different locations. This alignment task is addressed using a dynamic programming algorithm, DTW(dynamic time wrapping) [27].

In this chapter, we begin by discussing our approach to reconstructing the way-in path. Subsequently, we provide an overview of Dynamic Time Warping (DTW) and introduce a novel path-matching algorithm. This algorithm serves as the basis for a hybrid-matching system, which will be further explored in Chapter 6. It utilizes magnetic field data and steps/turns information to estimate users' position/orientation and detect statuses like off-route or reversed-route, facilitating path recovery. It is noted that the details of magnetic field data and its application to the path-matching algorithm will be further discussed in the next chapter.

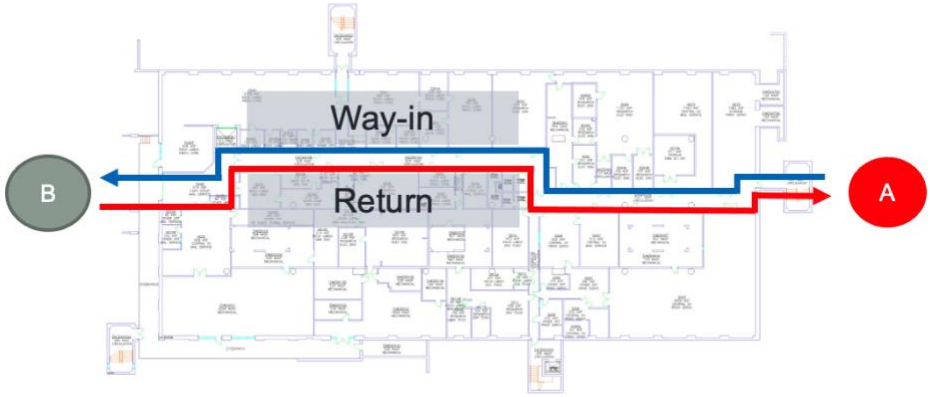


Figure 3.1: An example of assisted return. The blue line is the way-in path, and the red line is the return path. The way-in path starts from A to B, and the return path starts from B to A.

3.1 Path Reconstruction for the Way-in Route

As previously stated, our system does not rely on map information. However, it is essential to track the traveler to reconstruct the path during their way-in. For mapless navigation, SLAM (Simultaneous Localization and Mapping) techniques [41] are commonly used in robotics to reconstruct paths in real-time. Some pedestrian SLAM methods [42][43][44] have been proposed for mapping unfamiliar areas by leveraging sensor data collected from users who have traversed the same environment multiple times to reconstruct a real-time map. However, this approach isn't applicable to our system, as our app is designed to offer navigation for individual travelers who have visited a place only once and do not rely on prior knowledge from crowd-sourced data. In buildings characterized by networks of corridors, it is conceivable that walkers would proceed along relatively straight paths until they turn at a corridor junction. The angle made by two intersecting corridors for typical buildings is often equal to $\pm 90^o$. This geometric structural constraint can be leveraged to sidestep orientation drift. Following [21][29],we represent both way-in and return paths as a sequence of straight segments interleaved with discrete angle turns. We use the robust turn detector developed by a prior PhD student in our lab [12], which processes azimuth information using a Mixture Kalman Filter [45]. Although this algorithm was shown to work well even for multiples of 45^o turns, we constrained detection to multiples of 90^o for the purpose of this experiment (this reflects the type of junctions found in the building considered for our tests, which were all of $\pm 90^o$). The way-in path can be depicted as a 2-D polygonal chain (polyline), where the length of each segment is equal to the

number of steps taken in that segment, and consecutive segments have an angle as measured by the turn detector, as shown in Figure 3.2. Steps are detected using a LSTM network developed by another PhD student in our lab [46].

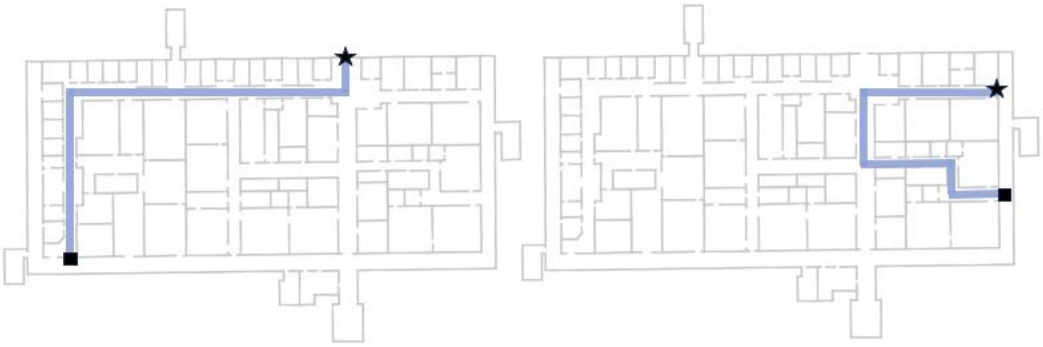


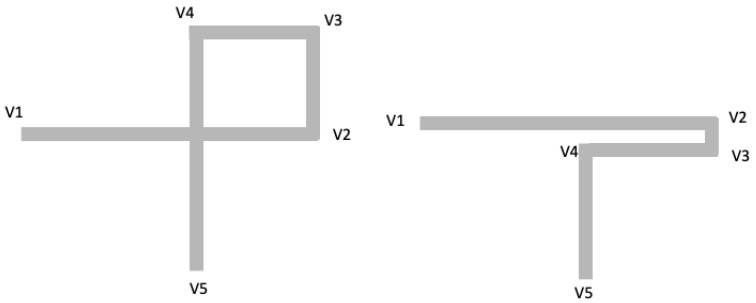
Figure 3.2: Examples of way-in paths depicted as 2-D polylines (thick blue lines) plotted on a building floorplan, with start points indicated by squares and endpoints by stars.

3.1.1 Path Simplification for the Way-in Route

In some cases, deviation from the initial way-in path may occur, resulting in additional unnecessary routes. These deviations could arise due to unintentional divergences from the correct path. Eliminating redundant paths before providing guidance for the return route is crucial for efficient navigation. While not all redundant paths can be removed without prior map information, some can be identified by examining the overall path, referred to as way-in simplification. These redundant paths fall into two categories:

1. **Paths with Closed Loops** (e.g., Figure 3.3(a)): The redundant paths form closed loops and can be identified as the path intersects itself.

2. **Paths with Open Loops** (e.g., Figure 3.3 (b)): These paths may consist of segments within the same corridor (in this example, $v2$ and $v3$ are the same location along the corridor but close to different sides of the wall), resembling extra steps, without forming closed loops.



(a) (b)

Figure 3.3: Two examples of additional paths. (a) paths with a closed loop; (b) paths with an open loop.

Let the original observed way-in path be represented as a 2D polyline graph, denoted as $G^{in} = (V^{in}, E^{in})$, where vertices V^{in} correspond to specific turn points along the route and edges E^{in} represent the straight segments connecting these points. The graph in Figure 3.3 (a) and (b) can both be defined as:

$V^{in} = \{v1, v2, v3, v4, v5\}$ and

$E^{in} = \{(v1, v2), (v2, v3), (v3, v4), (v4, v5)\}$,

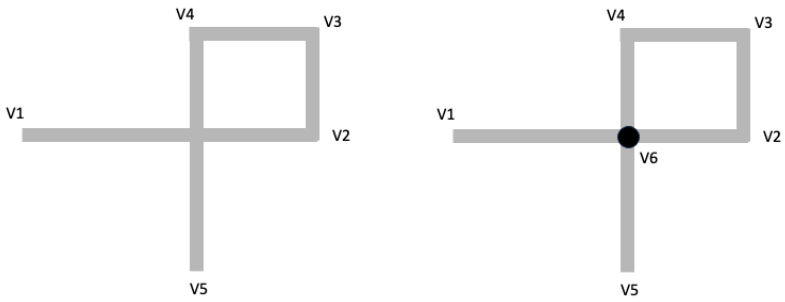
where the way-in path starts from *v*1 and ends in *v5*. The purpose of way-in simplification is to eliminate redundant loops. The methods to remove these loops, whether closed or open, in these cases are detailed as follows:

Paths with Closed Loops

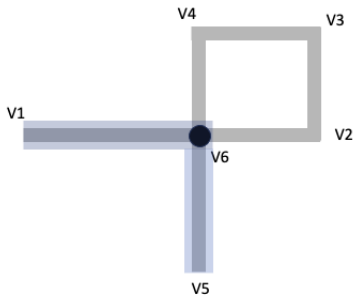
When encountering a closed loop, such as the one shown in Figure 3.4, an intersection is formed. By identifying these intersections, we can create additional vertices or nodes at these points. In this example, intersections between segments are identified, resulting in the creation of extra vertex *v6,* as shown in Figure 3.4(b). The graph is updated by establishing directional connections between the new vertices and existing ones. The new graph representation in this example becomes:

$V^{in} = \{v1, v2, v3, v4, v5, \boldsymbol{v6}\}$

$E^{in} = \{(v1, v2), (v2, v3), (v3, v4), (v4, v5), (\boldsymbol{v1}, \boldsymbol{v6}), (\boldsymbol{v6}, \boldsymbol{v2}), (\boldsymbol{v4}, \boldsymbol{v6}), (\boldsymbol{v6}, \boldsymbol{v5})\}$



(a) (b)



(c)

Figure 3.4: An illustration of simplifying the way-in path by removing a closed loop (a) The detected way-in path with a closed loop (b) An intersection $v6$ is identified between edges $(v1, v2)$ and$(v4, v5)$. (c) The shortest path from the start point $v1$ to endpoint $v5$ is shown in blue.

After updating the graph with additional vertices and edges, such as adding $v6$ in Figure 3.4(c), algorithms like Dijkstra's Algorithm can be applied to find the shortest path. In this example, the shortest path from the starting point $v1$ to endpoint $v5$ in the updated graph is $v1 \rightarrow v6 \rightarrow v5$.

Additionally, it is crucial to maintain a record of the length and direction of each edge whenever the graph is updated. This information ensures the consistency of the relative positions between the vertices.

Paths with Open Loops

In the scenario depicted in Figure 3.5, the walker misses the turning point $v4$ and continues walking straight. Assuming it happens in the same hallway, in order to go back in the right direction, they execute a 90^o turn at $v2$, proceed for a few steps, and then make another 90^o turn at $v3$, resulting in a U-turn within the same area.

Another potential scenario is that a wall blocks the walker from making a right turn, as illustrated in Figure 3.6 (the brown area indicates the wall). However, this obstruction results in a significantly larger distance D between the edges $(v1, v2)$ and $(v3, v4)$. Consequently, our method can detect this situation(as discussed in the following paragraph), ensuring that such situations are not mistakenly simplified.

In the case shown in Figure 3.5, where different segments can be created within the same corridor. Identifying parallel edges (in this example, $(v3, v4)$and part of $(v1, v2)$) can help us recognize paths within the same corridor. To address this, we first ensure that the distance between parallel edges is smaller than a threshold value denoted as r. If this condition is met, we proceed to merge these edges as follows. Taking the parallel edges $(v1, v2)$ and $(v3, v4)$ as an example, we then examine whether the vertices of these edges can be projected onto each other. As shown in Figure 3.5(b), $v4$ can be projected onto the line segment $(v1, v2)$, allowing us to create a new vertex, $v6$. Additional edges are subsequently formed on this segment, specifically $(v1, v6)$ and $(v6, v2)$, resulting in the updated graph:

$V^{in} = \{v1, v2, v3, v4, v5, \boldsymbol{v6}\}$

$E^{in} = \{(v1, v2), (v2, v3), (v3, v4), (v4, v5), (\boldsymbol{v1}, \boldsymbol{v6}), (\boldsymbol{v6}, \boldsymbol{v2})\}$

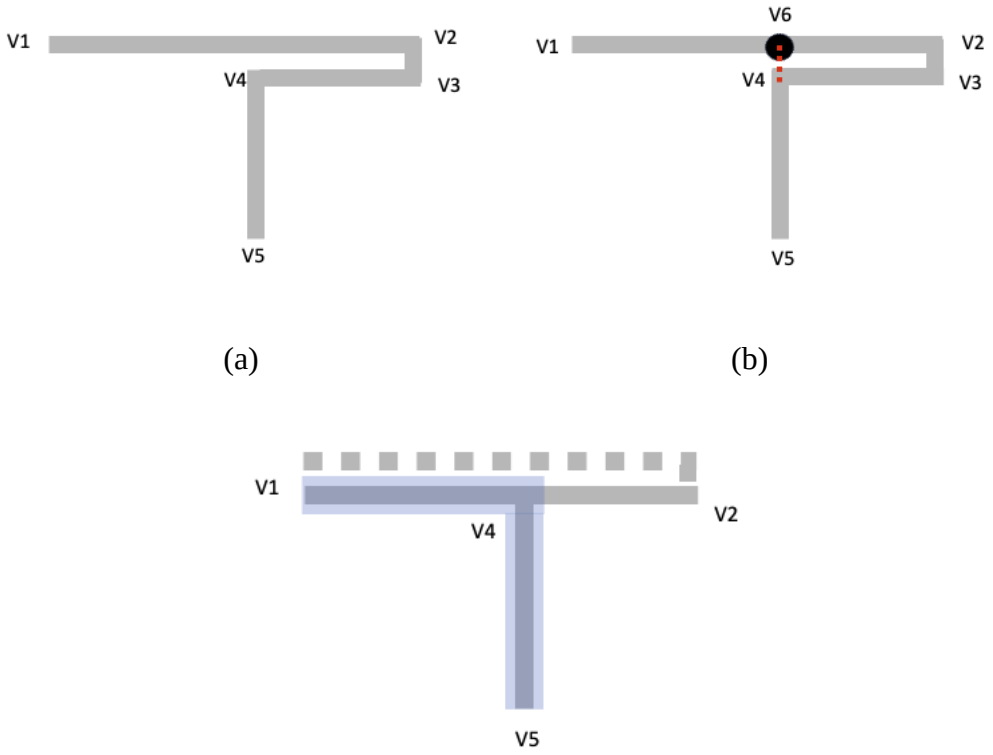


Figure 3.5: An illustration of simplifying the way-in path by removing an open loop (a)detected way-in path with an open loop (b) a projected point $v6$ is identified from $v4$ to segment $(v1, v2)$ (c) The shortest path from the start point $v1$ to endpoint $v5$ is shown in blue. The shaded path is the eliminated edges after way-in simplification.

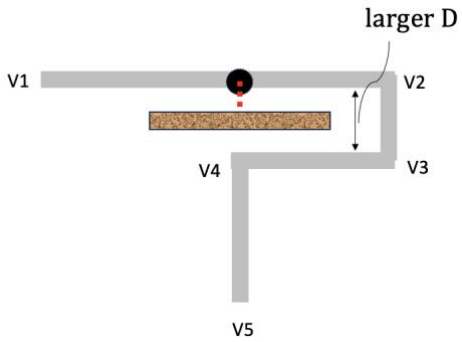


Figure 3.6: A situation where an open loop should not be simplified, the brown area represents a wall blocking the direct path from the turning point (black dot). However,

this obstruction results in a larger distance D, which will not be simplified by our method.

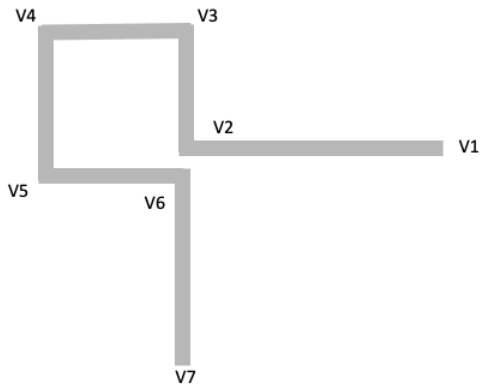


Figure 3.7: Another example of a redundant path with an open loop. The nearby vertices $v2$ and $v6$ can be merged.

After checking for parallel edges, we then examine all vertices to identify any close pairs, meaning vertices with a distance smaller than the threshold r. Despite already checking for the distance between processed parallel edges, we reevaluate all vertices to account for potential scenarios involving other segments (e.g., in Figure 3.7, vertices $v2$ and $v6$ are near each other, suggesting they may belong to the same point). In the example in Figure 3.5, vertices $v4$ and $v6$ are close to each other, indicating they may belong to the same location along the corridor. Similarly, vertices $v2$ and $v3$ are also close. Since the indices were labeled in temporal order, with the walker starting their path from $v1$ and then proceeding to $v2$, $v3$, and subsequent vertices, we chose to merge the vertices by replacing the one with the larger index number with the one having the smaller number. This decision is based on the assumption that a walker

reaches a smaller vertex index earlier during traversal of the way-in route; thus, likely to accumulate fewer mistakes on the path up to that vertex. While this assumption may not always hold true, this simplifies the problem without extra calculation on the coordinate of the merged vertex. The updated graph for the example in Figure 3.5, therefore, becomes:

$V^{in} = \{v1, v2, v5, \boldsymbol{v6}\}$

$E^{in} = \{(v1, v2), (\boldsymbol{v2}, \boldsymbol{v6}), (\boldsymbol{v6}, \boldsymbol{v5}), (\boldsymbol{v1}, \boldsymbol{v6}), (\boldsymbol{v6}, \boldsymbol{v2})\}$

(Note: duplicated vertices are removed)

We can utilize Dijkstra's Algorithm to find the shortest path from $v1$ to $v5$, resulting in $v1 \rightarrow v4 \rightarrow v5$. Like the closed loop scenario, we consistently record the length and direction of each edge during the update process of the polyline graph. This ensures that the new polyline maintains the correct relative position between the vertices after updating the graph.

To summarize, for real-time navigation, we propose a systematic method to handle both scenarios at the same time by the following procedures:

1.**Observing Intersecting Edges**: Compare all edges in the graph. If there are intersections between edges, create vertices based on these intersections and update the edges/vertices in the graph, following the first case described earlier.

2.**Managing Parallel Edges**: Identify parallel edges, and if the distance D between them is smaller than a predefined threshold r, create additional vertices based on the second case outlined earlier. Update the edges/vertices in the graph accordingly.

3.**Merging Close Vertices**: Compare the distance between any two vertices. If the distance between them is smaller than a predefined threshold r, consider the vertices as a single vertex and merge them while preserving the one with the smaller index. Update the graph accordingly.

4.**Finding the Shortest Path**: Utilize Dijkstra's Algorithm to determine the shortest path from the starting point to the destination.

The threshold r in the application is set to 7 steps, corresponding to 3.5 meters, given an average step length of 0.504 meters (the value is further discussed in the user study in Chapter 7). We expect corridors to generally be less wide than 7 steps to reduce the risk of mistakenly simplifying a wall belonging to the same corridor (the minimum width of corridors in California's commercial buildings[47] is estimated at around 2.4 meters, which is equivalent to about 5 steps).

3.2 Review: DTW and iDTW

The main objective of the system is to match the real-time information obtained during the return phase with that collected during the way-in phase, thus providing positional information to users based on the matched way-in location. This problem can be addressed by aligning two time sequences, for example, magnetic field data from the way-in and return paths. The goal is to synchronize them even in the presence of variations in sampling rates, lengths, or disturbance. Dynamic Time Warping (DTW) is a dynamic programming algorithm used for this purpose [27],[28]. DTW creates a 2-dimensional grid known as the cost matrix, where each cell represents the accumulated cost (penalty) of aligning samples up to a specific point in both sequences. By iterating through this grid and finding the path with the minimum total cost, we can determine the best alignment between the two sequences. Figure 3.8 provides an illustrative example of the sequence alignment by DTW.

In real implementation, given two time sequences $X^{in} = \{x_1, x_2, \dots, x_n\}$ and $Y^{ret} = \{y_1, y_2, \dots, y_m\}$, we can build a 2-dimensional cost matrix based on the following equation.

$$C(i,j) = \min\big(C(i-1,j-1), C(i,j-1), C(i-1,j)\big) + \left|x_i - y_j\right|$$

where $C(i,j)$ is the element of cost matrix for $i-th$ index in X^{in} and $j-th$ index in Y^{ret}. The optimal matched sequence can be determined by inspecting the minimum element of the last row in the cost matrix and then tracing back the path to the beginning of the matrix $C(1,1)$. Figure 3.9 is an example of matching the magnitude of magnetic field vector by DTW. Note that details of magnetic field are provided in Chapter 4.

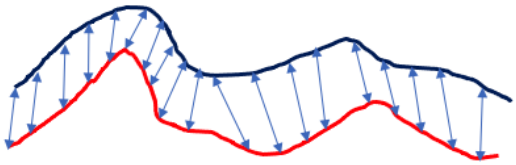


Figure 3.8: The alignment of two time sequences, blue and red lines.

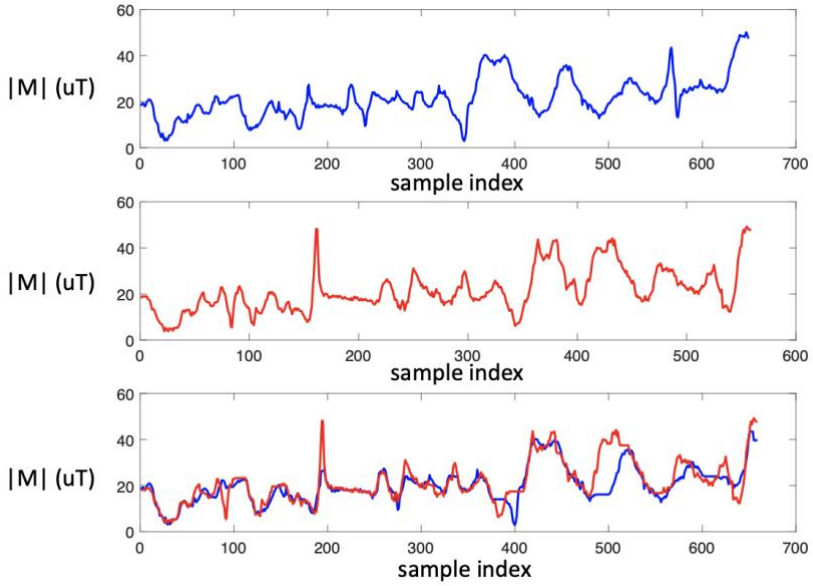


Figure 3.9: Matching the magnitude of magnetic field before and after DTW. Top: the blue line is the magnitude of magnetic field recorded in a path at an earlier time with known positioning information, called the "way-in" path. Middle: The red line is the magnitude of magnetic field collected later, called the "return" path. Bottom: aligned way-in and return sequences after applying DTW (blue: way-in; red: return).

In our application, we match the return sequence to the way-in sequence. To achieve real-time sequence matching, for every time instant on the return path, we have to update one column in the cost matrix (the cost matrix has a size of $m \times n$). For example,

at time instant j, the whole $j-th$ column of the cost matrix needs to be computed. When the length of the way-in sequence (m) is large, this computation can be costly. Therefore, a revised DTW method, called incremental DTW (iDTW), was proposed by Timothy et al. to reduce the computation complexity [18]. iDTW used a real-time approach with a sliding window to determine the current best-matched sample on-the-fly. At every time instant j, only a portion of the $j-th$ column in the cost matrix is calculated based on a defined sliding window. The sliding window is defined by the starting index of the window, $C_{j-1} - \frac{window\_size}{2}$, and the last index of the window, $C_{j-1} + \frac{window\_size}{2}$, where C_{j-1} is the minimum element in the previous column in the cost matrix.

This method does not require computing the overall cost matrix. However, the optimal path might be overlooked because it assumes that the best mapping for the current sample y_j is located within a range of the sliding window based on the last best mapping on sample y_{j-1}. This assumption is not necessarily true in real implementation so a proper window size should be selected for iDTW.

3.3 Path-Matching Algorithm

Our path-matching algorithm utilizes a combination of magnetic field and inertial data, specifically steps and turns, as inputs. During return, the user is assumed to start from the endpoint of the way-in path and walk the same path in the reverse direction. The

goal is to identify the location in the way-in path that best matches the current location (during return) of the user so that appropriate directions can be provided.

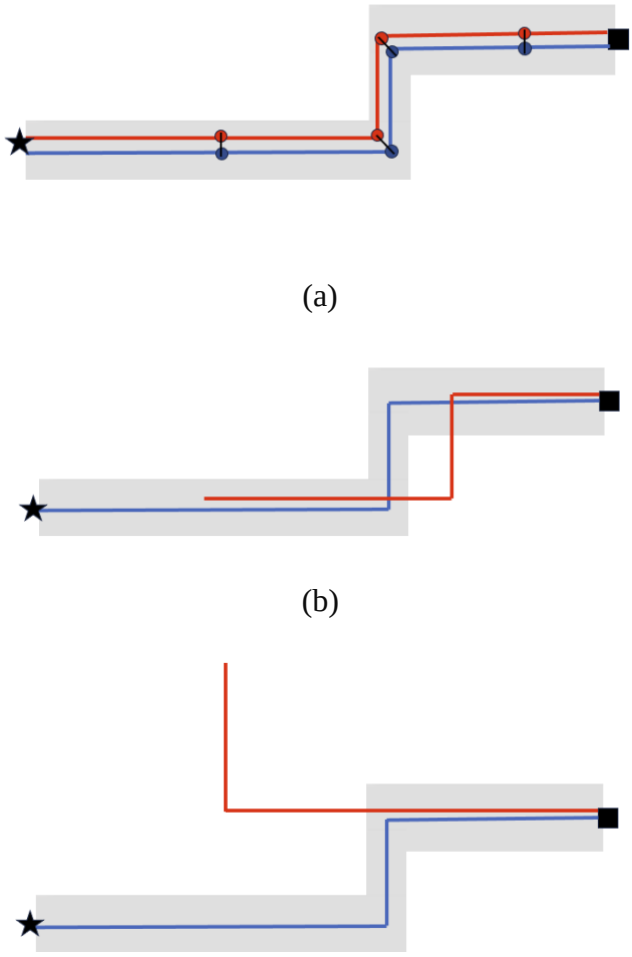
Our strategy for matching the return path with the way-in path is based on the coordination of two different algorithms: **projected return sequence** and **sequence alignment**.

Projected Return Sequence

This algorithm creates a polyline to represent the return path, as described previously for the way-in path. By comparing the current polyline reconstructed during the return with the polyline we built during the way-in, it is possible to find the best match to where the user is now. For instance, we can identify the closest point on the way-in polyline to the user's current position by calculating their respective distances. This process is illustrated in Figure 3.10(a), where the matched positions for the way-in and return paths are depicted by blue and red dots, respectively.

In theory, if accurate odometry data can be stored, one could efficiently match the current estimated position during the return phase with the closest position along the way-in path by aligning the turns or steps in both phases. However, this method faces challenges in real-life situations, as illustrated in Figure 3.10(b) and (c), where the length of a segment equals the number of steps. For instance, the user's steps may vary in length between the way-in and return journeys (Figure 3.10(b)); there may be instances where turns are missed or falsely detected(Figure 3.10(c)). All of these situations may be expected, especially when someone walks without visual feedback.

These challenges highlight the importance of implementing sequence alignment algorithms to improve overall performance.



(a)

(b)

(c)

Figure 3.10: Examples of real-life challenges in the way-in and return paths. Red lines: projected return trajectory. Blue line: way-in trajectory. The start and end points for the way-in path are indicated by a square and a star, respectively. In this simplified representation, the length of each segment is equal to the number of steps (a) In an ideal situation, the path can be easily matched by finding the closest positions in the way-in and return polylines. The matched positions are shown in blue and red dots, respectively. (b) The walker takes a longer step length during return, resulting in fewer steps and shorter lengths in each segment of the polyline. (c) The first turn during the return was not detected.

Sequence Alignment

When matching the current position during the return journey with the previously acquired (reversed) way-in path, as described in [18][29][48], can be approached through (sub)sequence alignment, with the concept of a graph formalized as follows.

Firstly, the sequence of way-in measurements is reversed, which is convenient since the route is being backtracked. At each time during return, we aim to determine the initial way-in subsequence of measurements that best matches the current sequence of return measurements. Symbolically, given the (reversed) way-in sequence W of measurements (observations) $o^{in}(t)$ (i.e., $W = (o^{in}(1), \ldots., o^{in}(N))$, and the current sequence R of return measurements $o^{ret}(t)$ (i.e., $R = (o^{ret}(1), \ldots., o^{ret}(J))$, the goal is to find a sequence of indices $i_1, \ldots., i_J$ such that $(o^{in}(i_1), \ldots., o^{in}(i_J))$ best matches R under an appropriate criterion.

For real-time guidance, particularly at return time index J, we are interested in the last matching point i_J during way-in as it represents the best-mapped position computed based on the return data up to the current time index. Dynamic Time Warping (DTW), as described in section 3.2, can be utilized to find an optimal match.

Regarding the considered measurements—magnetic field vectors, detected turns, and steps—step detection is implicitly accounted for: both in the way-in and return phases, the sequences of time indices are structured such that there are three regularly spaced time intervals between two consecutive detected steps. This choice provides sufficient spatial granularity for magnetic field matching while ensuring efficient sampling, with no samples recorded when the user is stationary. As mentioned in the previous section,

in theory, if accurate odometry data can be restored, we can easily map the walker's position. For example, if someone walked 100 steps and then turned right during the (reversed) way-in, it is expected that in the return path, they will also walk approximately 100 steps and then turn right, with the magnetic signatures hopefully matching. However, in real-life situations, this may not hold true; for example, the walker may take a different number of steps in the same segment in the way-in and return paths, or the walker may miss a turn during the return path. As a result, the path-matching algorithm is proposed to provide better mapping.

Given these measurements, a directed graph $\mathcal{G}$ can be constructed with nodes indexed as (i, j), where i represents a way-in time index and j represents a return time index. It's important to highlight that $\mathcal{G}$ is constantly expanded as the walker progresses along the return path. As shown in Figure 3.11, each node (i, j) in the graph has three edges: to $(i+1, j)$, $(i+1, j+1)$, and $(i, j+1)$, respectively. This configuration aligns with the assumption that the walker typically moves in the same direction as in the (reversed) way-in path but possibly with different step lengths, resulting in steps detected in either phase that cannot be matched in the other phase, which is accounted for by the edges to $(i+1, j)$ and$(i, j+1)$. The edges to $(i+1, j)$, and $(i, j+1)$ carry an edge cost, named non-diagonal cost, indicating a penalty if two time instants are matched to the same time instant in the other path. There are different node costs introduced based on input observations, which are the cost of magnetic field (C_{MF}) and the cost of unmatched turns, as detailed below:

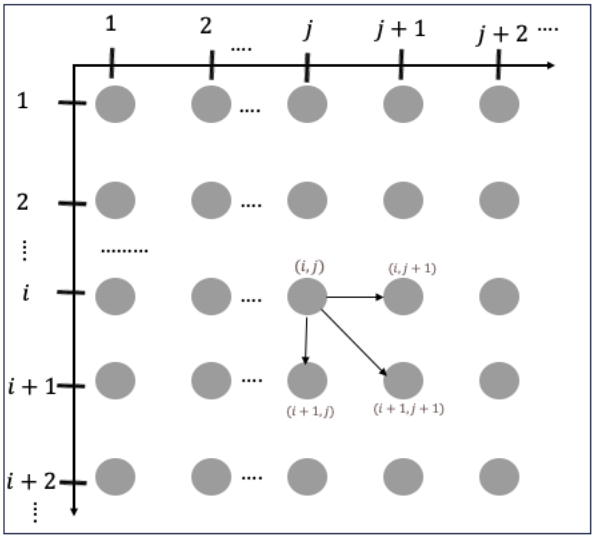


Figure 3.11: The connections of node (i, j) in graph $\mathcal{G}$. x-axis in the sample indices during the return and y-axis in the sample indices during the way-in.

3.3.1.1 Node cost of discrepancy in magnetic field (C_{MF})

We defined the node costs, cost of magnetic field (C_{MF}) as a function of the discrepancy in the magnetic field between measurements $o^{in}(i)$ and $o^{ret}(j)$. Given the expectation that the same location should exhibit similar magnetic fields, a higher discrepancy in magnetic field results in a higher cost of magnetic field C_{MF}. Figure 3.12 shows an example of magnetic field discrepancies for all nodes in the graph $\mathcal{G}$. A detailed description of C_{MF} is provided in Chapter 4.3.

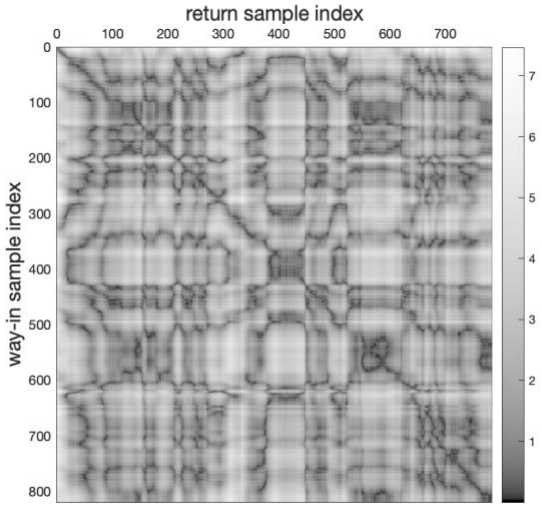


Figure 3.12: An example illustrating magnetic field discrepancies (intensity bar displayed on the right, unit in μT) for all nodes in the graph $\mathcal{G}$.

3.3.1.2Node cost of unmatched turns (C_{UT})

Another node cost C_{UT} (cost of unmatched turn) is introduced when the walker makes a turn. If, for example, at time index i, a 90^o turn was taken during way-in traversal, one could expect that, at the matching time index j, a turn by -90^oshould be observed during way-in. A simple way to leverage this intuition is to define, at each way-in turn (index $\bar{i}$), a certain node cost C_{UT} for all nodes $(\bar{i},:)$ in the graph, except for the nodes $(\bar{i},j)$ in which a turn by the opposite angle was observed during return. Likewise, a turn at $\bar{j}$ during return would determine a node cost for all nodes $(:,\bar{j})$ for which a way-in turn by the opposite angle was not observed. These node costs could be added to the node costs defined for magnetic discrepancies, encouraging the optimal path to include matching turns at way-in and return.

With this definition of graph $\mathcal{G}$, the sequence alignment is obtained by finding the minimum cost path originating from $(0,0)$, and terminating at a node (i,J). However, this simple approach is liable to fail in common situations with short sequences of incorrectly detected turns. For example, suppose that during the (reversed) way-in, someone walked 100 steps and then turned right (-90^o turn). During the return, it is expected to detect approximately 100 steps and a right turn. However, at the turn junction, the walker stopped briefly, turned their body to the left (perhaps to respond to a greeting from a passerby), then resumed walking and made a right turn. In this case, the system may incorrectly detect a left turn (90^o turn) followed by a 180^o turn. Correctly matching the way-in and return paths in such cases may result in two unmatched turns. Moreover, one of the spurious turns in the return path may erroneously match with some other distant turn during the way-in, potentially creating a significant path mismatch.

This example suggests that an unmatched turn should be given a lower penalty when preceded shortly by another turn and the accumulated turn angle (in this example, $90^o + 180^o = 270^o$, i.e., -90^o turn during the return phase) is the same as the expected turn angle (in this example, -90^o turn during the way-in phase). However, this cannot be implemented by simply assigning costs to edges or nodes in our original graph. To address this challenge, we propose a mechanism that considers the orientation of the walker at each time, obtained by accumulating detected turns. This idea is described as follows.

This approach organizes the previously defined graph $\mathcal{G}$ into a series of layered planar graphs, with each layer representing a possible orientation discrepancy between the way-in and return paths. By "orientation discrepancy," we mean the angular difference between the walker's measured orientation at some time j during return and the opposite of the orientation of the walker measured at some time index i during way-in. For example, if in the return phase, a walker takes two consecutive 90^o (left) turns followed by a -90^o (right) turn, their orientation at that point will be $90^o + 90^o - 90^o = 90^o$ (as defined with respect the starting walking direction). Ideally, the orientation discrepancy should be consistently equal to 0^o for a correctly matched sequence, though this may not be the case when a turn is missed by the detector, or false turns are detected, or the walker takes a detour.

A node is now indexed by the triplet (i, j, d) , where layer d (for $0 \leq d \leq 3$) represents an orientation discrepancy of $d \cdot 90^o$, as shown in Figure 3.13. In our case, since there are only 4 possible orientations (as only turns by multiple of 90^o are allowed), there are 4 possible discrepancies $(0^o, 90^o, 180^o, -90^o)$. The different graph layers have identical topology and edge costs. In particular, a node (i, j, d) is connected to nodes $(i + 1, j, d)$, $(i + 1, j + 1, d)$ and $(i, j + 1, d)$ in the same layer. The node costs at layer d are the sum of the magnetic discrepancy costs C_{MF} (identical across layers) and of constant mis-orientation cost C_{mo} for layers with non-zero orientation $(d \neq 0)$. This cost discourages long paths with non-zero orientation discrepancy. If a turn by $k \cdot 90^o$ is detected during way-in at time $\bar{i}$, additional edges are created between $(\bar{i}, j, d)$ and node $(\bar{i} + 1, j, (d - k)\ mod\ 4)$ and $(\bar{i} + 1, j + 1, (d - k)\ mod\ 4)$ for all j

and d except for those j in which a turn by $-k \cdot 90^o$ was detected. These edges account for the fact that a way-in turn at $\bar{i}$ that is unmatched with a return turn at j modifies the walker's orientation discrepancy. The original edges from $(\bar{i}, j, d)$ to nodes $(\bar{i}+1, j, d)$ and $(\bar{i}, j+1, d)$ (same layer) are maintained, but with a higher associated edge cost C_{UT} (cost of an unmatched turn). A path in the graph going through either such edge indicates that this detected way-in turn has been "rejected" (since the orientation discrepancy has not changed), as shown in Figure 3.14. It is noted that the edge between $(\bar{i}, j, d)$ and $(\bar{i}+1, j, (d-k)\ mod\ 4)$ is not created because the same turn by $k \cdot 90^o$ was detected both of these nodes so the walker should not have orientation discrepancy between these two nodes. Likewise, as shown in Figure 3.15, a turn by $-k \cdot 90^o$ detected at time $\bar{j}$ during return generates new edges from $(i, \bar{j}, d)$ to $(i, \bar{j}+1, (d-k)\ mod\ 4)$ and to $(i+1, \bar{j}+1, (d-k)\ mod\ 4)$ (unless a turn by the opposite angle was detected at i), while the cost of edges C_{UT} from $(i, \bar{j}, d)$ to $\left(i, \bar{j}+1, d\right)$ and $(i+1, \bar{j}+1, d)$ applies. The interplay between orientation discrepancy costs and "turn rejection" costs helps deal with incorrectly detected turns or with situations in which the walker, during return, briefly detours from the way-in path.

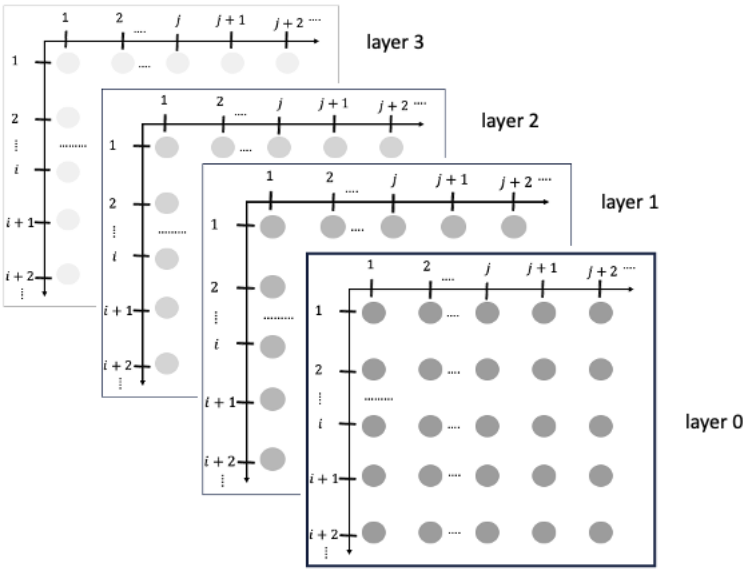


Figure 3.13: The layered graph. The x-axis represents the sample indices in the return route, while the y-axis represents the sample indices in the way-in route.

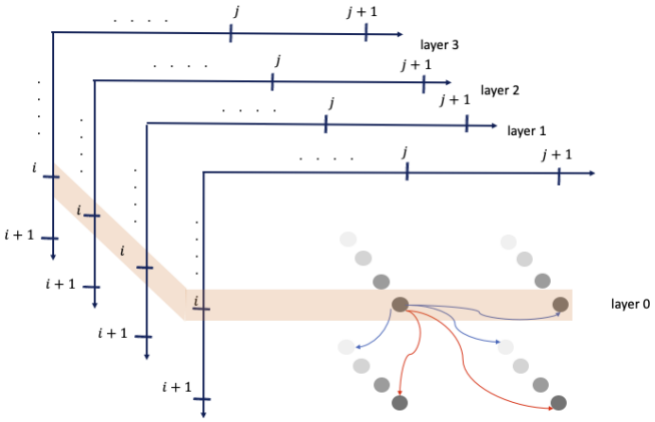


Figure 3.14: Edge connections for node $(i, j, 0)$ between layers when a turn $(k = 1)$ occurs during way-in.

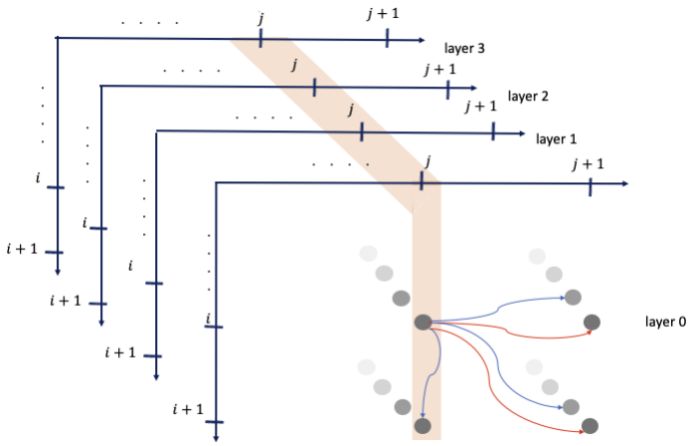


Figure 3.15: Edge connections for node $(i, j, 0)$ between layers when a turn $(k = -1)$ occurs during return.

The minimum cost path in the graph $\mathcal{G}$ is recomputed at each new return sample. We use the incremental Dynamic Time Warping (iDTW) [riehle2012indoor], which uses a sliding window defined around the endpoint of the previously found optimal path (we set the window size equal to 300 samples). Although this algorithm produces a suboptimal solution, particularly if the best mapping for the current sample falls outside the sliding window range, it still represents a good compromise between precision and computational cost.

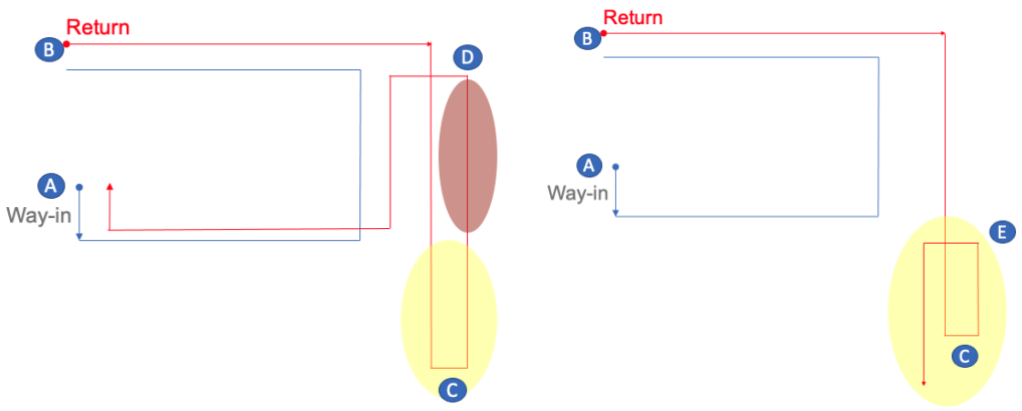
3.4 Off-Route and Reversed-Route Detection

While a visually impaired walker tries to transverse their path with the assisted return system, there are two challenges in the task. The first challenge arises when the walker deviates from the intended route. This deviation may occur due to a missed turn (Figure 3.16, yellow shaded path) or taking a turn too early (e.g., counting the incorrect number

of steps before the next turn). To address this issue, we propose a modified path-matching algorithm capable of detecting when users are off-route. The second challenge arises when the system must guide the user back onto the correct path after they have been identified as off-route.

For example, in the case shown in Figure 3.16 (a), the users were instructed to make the first U-turn at location C and return to a previously detected on-route location. Subsequently, they made another U-turn to rejoin the path at location D. However, providing specific step and turn instructions for users to backtrack may lead to inaccuracies, as users' step lengths can vary. Consequently, users may remain off-route (Figure 3.16(b), location E) despite following the system's directions intended to guide them back onto the route.

To address this, we propose a mechanism to make sure that after the user makes the first U-turn from the off-route status, the user is positioned correctly on the path but facing the opposite direction. This ensures that the system accurately determines the user's position before issuing further guidance. Consequently, we introduce the concept of the reversed-route status, depicted by the brown shaded path in Figure 3.16(a). This status indicates that the user is on the correct path but facing in the opposite direction, typically occurring after the user has been detected as off-route and attempts to retrace their steps. The algorithm supporting off-route and reversed-route detection is described in the following sections.



(a) (b)

Figure 3.16: Path recovery after off-roue is detected. The return path starts from B to A. (a) Off-route (yellow path) and the following reversed-route (brown path) are properly detected. Path recovery is successfully executed in this scenario. (b) Without the mechanism of detecting a reversed-route status, the user makes the second U-turn too early and is unable to be back on the correct path.

3.4.1 Off-Route Detection

When the walker is not on the correct path, the steps/turns data has limited information to identify the off-route status. For example, taking extra steps does not necessarily mean being off-route because the walker might have different stride lengths between the way-in and return paths. Another similar example is that the system might detect a spurious turn because the user changes the smartphone's position. Therefore, the magnetic field plays an important role in detecting the off-route status by comparing the signature of the magnetic field between the on-route path and the off-route path. In other words, when the detected magnetic field on the return path is similar to it on the way-in path, the chance that the user is off-route is low and vice versa. To incorporate this concept into our algorithm, we introduce a fifth layer into the previously defined

graph, specifically designated as the off-route layer. This is achieved by introducing a triplet, (i, j, d) where $d = 4$, indicating the off-route layer.

A path through a node $(i, j, 4)$ in the off-route layer indicates a high chance of the walker being off-route. Consequently, less penalty ($C_{MF_{off}}$, cost of magnetic field for off-route layer) should be applied to the node when there is a higher discrepancy in the magnetic field between the mapped pair (i, j). In contrast, for regular nodes (layer $d \neq 4$), the penalty (C_{MF}, cost of magnetic field) is high when there is a higher discrepancy in the magnetic field. Consequently, $C_{MF_{off}}$ demonstrates an inverse relationship with C_{MF}, such that as $C_{MF_{off}}$ increases, C_{MF} decreases, and vice versa. This relationship allows us to utilize the C_{MF} to identify the $C_{MF_{off}}$ for the off-route layer.

The node cost of the off-route node is defined by $C_{MF_{off(i,j)}}$. This is computed as:

$$C_{MF_{off(i,j)}} = \max\{(mag_{thres} - \alpha \cdot C_{MF(i,j)}), 0\}$$

Where:

mag_{thres} is the threshold of the cost of magnetic field for the off-route node.

$C_{MF(i,j)}$ is the cost of magnetic field for the on-route nodes as defined in section 3.3.1.1.

α is a parameter that controls the increasing rate of $C_{MF_{off(i,j)}}$ when $C_{MF(i,j)}$ decreases.

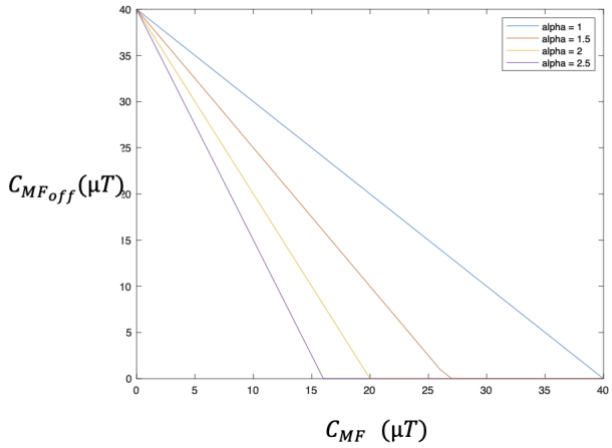


Figure 3.17: Relationship between $C_{MF_{off(i,j)}}$ and $C_{MF(i,j)}$. In this example, mag_{thres} is set to 40.

As mentioned earlier, when there is a higher discrepancy in the magnetic field between the mapped positions (i.e., when $C_{MF(i,j)}$ is large), it implies that the chance of the user being off-route is high. In such case, $C_{MF_{off(i,j)}}$ is expected to be small. Figure 3.17 shows this relationship with different values of α.

Extra edges are created for nodes in the off-route layer. The nodes in layers 0 to 3 (on-route layers) have direct edges to the off-route layer, as shown in Figure 3.18. There are two distinct groups of edges:

The first group of edges starts from the on-route layers. These edges indicate that one can be on the correct path but become off-route in the next time instant. Such transitions typically occur when $C_{MF_{off(i,j)}}$ is small (or large $C_{MF(i,j)}$). However, temporary fluctuations in the magnetic field (e.g., due to a running elevator) might erroneously

trigger this transition. To mitigate this, a high edge cost is assigned to this group, denoted as $C_{sts1_{off(i,j)}}$.

The second group is associated with the connections from the nodes in the off-route layer. Consequently, a smaller penalty, $C_{sts2_{off(i,j)}}$, is applied to the edges.

The additional edges for the nodes in the off-route layer are shown in Figure 3.18.

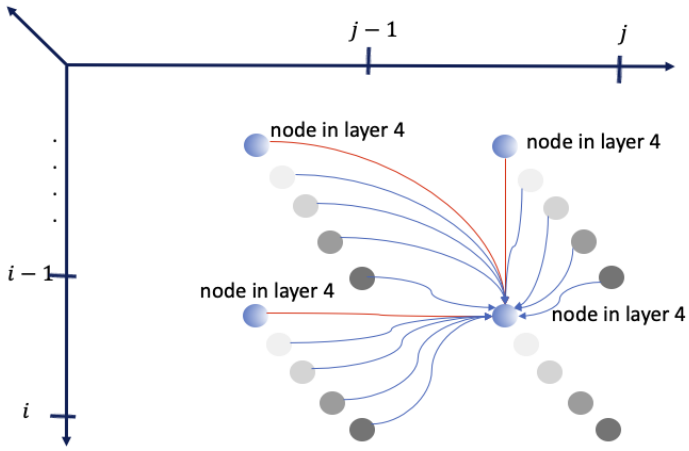


Figure 3.18: Additional connections to the node $(i, j, 4)$ (layer 4) in the off-route layer. Blue nodes indicate nodes in the off-route layer (layer 4). $C_{sts1_{off(i,j)}}$ applies to blue edges and $C_{sts2_{off(i,j)}}$ applies to the red edges.

3.4.2 Reversed-Route Detection

The concept of a reversed-route status occurs when the user is back on the correct path but facing in the opposite direction, resulting in an orientation discrepancy of 180^o. Therefore, one of the previously defined layers (specifically layer 2 with $d = 2$, where the orientation discrepancy equals 180^o) can be utilized to detect reversed routes. In

this layer, a smaller cost assigned to a node indicates a higher likelihood that the walker was on a reversed route at the corresponding time instants associated with that node. Similar to the off-route detection process, the magnetic field contains valuable information for determining the reversed-route status. When a reversed-route occurs, the sampled sequences of magnetic field in the corresponding way-in and return paths resemble mirror images because they essentially represent the same magnetic field sampled in reverse order, as depicted in Figure 3.19.

When no turn is detected, a node $(i, j, 2)$ is connected to two sets of layers (Figure 3.20) .

- $(i-1, j, 2)$, $(i-1, j+1, 2)$, $(i, j+1, 2)$
- $(i-1, j, 4)$, $(i-1, j+1, 4)$, $(i, j+1, 4)$

The first set of connections assumes that the walker continues walking in reversed-route and the second set of connections represents the possibility that the user becomes off-route from reversed-route. A path with a decreasing time index in way-in and increasing time index in return sequence(e.g., $(i, j, 2) \rightarrow (i-1, j+1, 2)$ indicates that the incoming sample (on the return path) is mapped to a prior way-in sample (the way-in sample was already revered) which is similar to mapping the sample in a reversed order. Consecutive nodes in a path with a repeated index (e.g., $(i, j, 2) \rightarrow (i-1, j, 2)$) indicate that two time instants (in this example, during way-in) are matched to the same time instant in the other path. Note that an extra penalty $C_{sts_{rev}}$ is applied to the edges that connect the off-route nodes to the reversed-route node. This mechanism can

prevent a temporary change in the magnetic field (e.g., running an elevator or walking too close to the wall), triggering an unexpected detection on a reversed route.

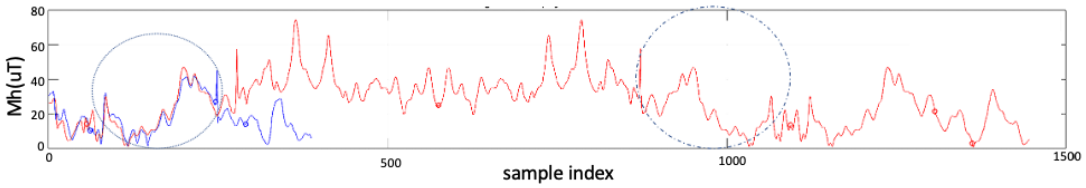


Figure 3.19: An example of the mirrored magnetic field signatures indicating reversed-route. Blue: magnetic field detected during the way-in. Red: magnetic field detected during the return. The magnetic field of the user being reversed-route is marked by the dotted-dashed ellipse, and the corresponding magnetic field in the way-in path is marked by the dashed ellipse on the left side of the figure.

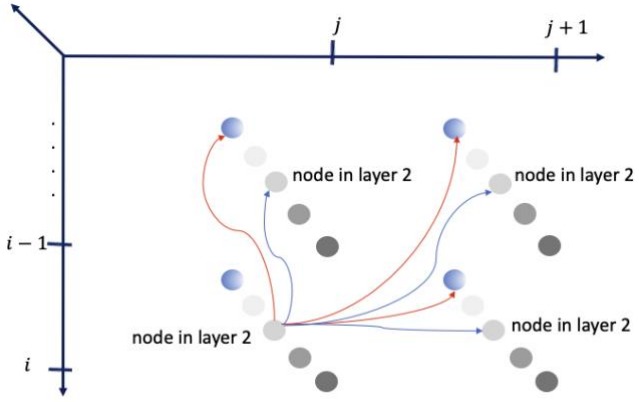


Figure 3.20: Extra edges from reversed-route node$(i, j, 2)$. $C_{sts_{rev}}$ applies to the red edges.

When a turn by $k \cdot 90^o$ is detected at time j ,there are extra edges created for node $(i, j, 2)$ connecting to 4 sets of nodes (Figure 3.21):

- $(i-1, j+1, (2-k)\ mod\ 4)$, $(i, j+1, (2-k)\ mod\ 4)$
- $(i-1, j+1, 2)$, $(i, j+1, 2)$.
- $(i-1, j, 2)$

- $(i-1, j, 4)$, $(i-1, j+1, 4)$, $(i, j+1, 4)$

The first two connections are made under the assumption that the turn was correctly detected, which causes an update on the difference of the walker's orientation between way-in and return. The second set of connections represent the possibility that the turn was incorrectly detected, meaning that the orientation discrepancy should not be changed. The "turn suppression" cost C_{ts} is applied to the edges. The third connected (from $(i, j, 2)$ to $(i-1, j, 2)$) indicates that both time instant i and $i-1$ are matched to j. The decision of whether to accept this turn is postponed till node to $(i-1, j+1, 2)$ or $(i-2, j+1, 2)$, and therefore the turn suppression cost C_{ts} is not applied here. The fourth set of connections indicates the possibility of going off-route from a reversed-route.

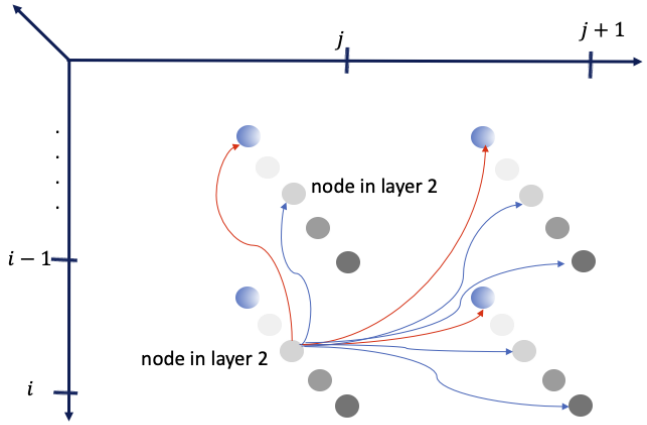


Figure 3.21: In this example, a 90^o turn was detected at the time instant j, and extra edges are created for node *(i,j,2)*. $C_{sts_{rev}}$ applies to the red edges.

3.5 Error Metrics

To evaluate the correctness of path matching, it would be necessary to access the "ground truth" matches – i.e., the correct sequence of matching time instant pairs of (i_j, j), where i_j is the mapped index for the return index j. However, this would require recording the user's position at every time instant during way-in and return routes, which requires extra tools, for example, a camera and SLAM (Simultaneous Localization and Mapping) algorithm to track the camera's position and orientation in real-time as it moves through the corridor. Instead, we propose an alternative error metric in this study that only records the time at which the walker transitioned between different segments, as shown in Figure 3.22. We interpolated the time matches between these discrete time/location data points by assuming that participants walked at a constant speed within each segment. This gives us an approximate location of the walker at all times. Based on this information, we can compute the set of nodes $\{\hat{l}_j, j, d\}$ that represent the correct matching of $\hat{l}_j$ with j (meaning that the walker was at the same location at time l_j during way-in as the walker at time j). When evaluating the correctness of a given node (i_j, j, d) in the graph path chosen by the path matching algorithm, we record the absolute difference between i_j and $\hat{l}_j$. This measures the error (in time instants) for node (i_j, j, d) . It's important to note that the orientation discrepancy d is not factored into the error calculation. This is because the mapped node may temporarily be in different layers even when the walker is in the same position due to variations in the timing of turn detection along the way-in and return

routes. The path-matching error is computed as the average error over the entire on-route path, as depicted in the following equation.

$$E_{PM} = \frac{\sum_{j=1}^{M} ||\hat{l}_j - i_j||}{M} \quad (3.1)$$

where M is the number of matched samples on the correct path.

The error metric E_{PM} helps us to determine the parameters of the basic path−matching algorithm. The values of the parameters are shown in Appendix A; they are established through multiple trials in initial experiments which generate smaller E_{PM}.

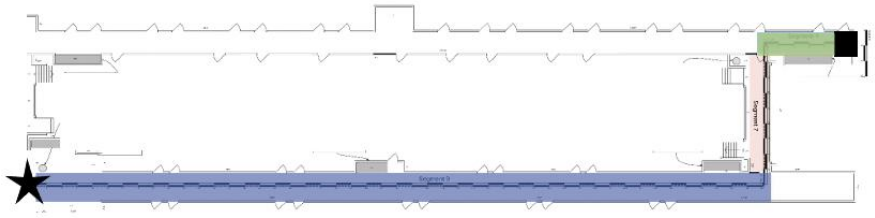


Figure 3.22: An illustration of segments to approximate the walker's location at each time. The start and end points for the path are indicated by a square and a star, respectively. There are three segments (blue, pink, and green) in the overall path. To approximate the walker's location at each time, the time that the walker entered/exited each segment was recorded and then interpolated by the location of the data points.

3.6 Conclusion

In this section, we address the challenges associated with leveraging magnetic field data and turns/steps information to backtrack a walker's position in situations when the map is unavailable. A straightforward approach to simplify the way-in route is introduced. Furthermore, a novel graph-based path-matching algorithm is presented, which considers variations in the walker's orientation between the way-in and return

routes by incorporating penalties into the cost matrix of the graph for different scenarios. Additionally, the algorithm addresses situations where the walker deviates from the intended path. To evaluate the performance of the algorithm, we defined an error metric for this basic path-match graph. Building upon this graph, we will propose a new method of hybrid matching in Chapter 6. The new method excludes the off-route layer, resulting in fewer parameters, which will be illustrated later.

Chapter 4

Path-Matching Algorithm: Magnetic Field

Magnetic field is increasingly used for indoor navigation due to its unique characteristics within indoor environments [9][22][23]. This navigation relies on a combination of the Earth's magnetic field and the presence of ferromagnetic objects indoors, resulting in distinct magnetic signatures that vary across locations. Smartphones equipped with magnetometers offer a convenient platform for implementing magnetic-based navigation solutions, providing advantages such as affordability, ease of use, and no infrastructure requirements. However, despite these advantages, challenges exist, including the need to pre-collect a map of the magnetic field in indoor spaces and dynamic magnetic interference from sources like running elevators and large electric appliances.

Our goal is to develop a mapless navigation system that utilizes real-time data for magnetic field analysis during the user's traversal of the path on their way-in route, thus eliminating the need for extensive offline collection of magnetic maps. Additionally, by employing a path-matching algorithm based on real-time analysis of the magnetic field and step/turn information (as described in Chapter 3), we aim to mitigate the effects of dynamic magnetic interference. The following section provides an in-depth

investigation of the magnetic field data, enabling the integration of this valuable information (specifically, the cost of magnetic field C_{MF}) with the path-matching algorithm.

4.1 1D/2D/3D Magnetic Field

Smartphones are equipped with 3-axis magnetometers that measure the 3D vector of the magnetic field, denoted as $\vec{M} = \{M_x, M_y, M_z\}$. While matching magnetic signatures, it's important to note that the reference frames of the magnetometers are not fixed, as they are continuously moving with the user's smartphone. One way to handle this changing reference frame issue is by using the norm of the magnetic field, also called the 1D magnetic field. The norm of the magnetic field represents its magnitude and remains independent of the reference frame. This method simplifies the magnetic field into a single value that shows how strong it is, no matter how the smartphone is held.

However, relying solely on the norm of the magnetic field may be insufficient, as multiple locations could exhibit the same values in the magnetic field's norm. To address this limitation, we can use the gravity vector $\vec{g}$ obtained from the accelerometers. The accelerometer's output combines both gravity and user acceleration, requiring the extraction of the gravity vector from the total acceleration. This task typically involves two additional sensors: the magnetometer and gyroscope. The magnetometer detects the Earth's magnetic field. Although it was influenced by the environment, it still can be utilized as a reference for determining the "downward" direction. Meanwhile, the gyroscope aids in distinguishing between user-induced

movements, and the constant pull of gravity. Through sensor fusion techniques[49], the smartphone distinguishes user acceleration from the total reading, isolating the gravity vector as a constant downward force. The gravity vector is expressed in the device's reference frame, which makes it possible to generate the 2D vector of the magnetic field; one is the magnitude of magnetic field's projection on the horizontal plane, and the other is the magnetic field in the gravity direction [37] [50]. The following equation gives the 2D magnetic field.

- $M_g = \frac{<\vec{M},\vec{g}>}{||\vec{g}||}$
- $M_h = \left(\left|\left|\vec{M}\right|\right|^2 - \frac{<\vec{M},\vec{g}>^2}{||\vec{g}||^2}\right)^{0.5}$,

where $\vec{M}$ is the 3D magnetic field in the device's reference frame, and $\vec{g}$ is the gravity vector in the device's reference frame. M_g is the magnetic field in the gravity direction, and M_h is the magnitude of magnetic field's projection on the horizontal plane.

The 3D magnetic field $\{M_x, M_y, M_z\}$ can also be used to match the magnetic signatures if one can get the accurate rotation matrix of the device with respect to a fixed reference frame. Indeed, it can even provide more precise positioning results because it preserves the three-dimensional magnetic data. However, an accurate rotation matrix relies heavily on inertial sensors. It is commonly known that inertial sensors are highly affected by bias, which can lead to drift. Therefore, the 2D magnetic field is mainly used in this study.

4.2 Magnetic Field's Characteristics

Subbu et al. investigated the magnetic field patterns due to the presence of furniture, elevators, doors, pillars, etc. [51]. It was discovered that the short-term displacement of an object (e.g., table, chair) does not cause a significant effect on the magnetic field. Furthermore, when a smartphone collects the magnetic field, the metallic objects in the user's pocket do not significantly affect the magnetic field. They also investigate the variance of magnetic field over time (7 months). As shown in Figure 4.1, there are no significant variations in the magnetic signatures, which could potentially diminish its effectiveness for sequence matching in localization

While using the magnetic field pattern for navigation, another factor that might influence the result is veering, an issue that a walker cannot keep walking straight [52]. It can cause variations in a magnetic field. As shown in Figure 4.2, when the user is walking in a hallway, the distance between the user and the wall leads to fluctuations in the magnetic field. This topic is further discussed in section 4.3.

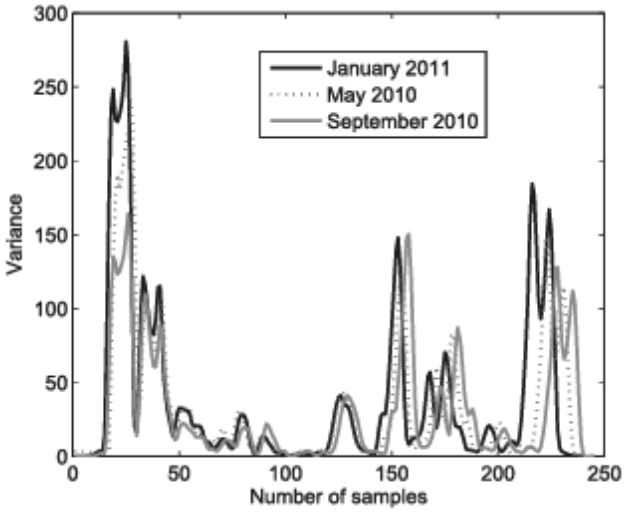


Figure 4.1: Prior research about the variance of the magnetic field over time [51].

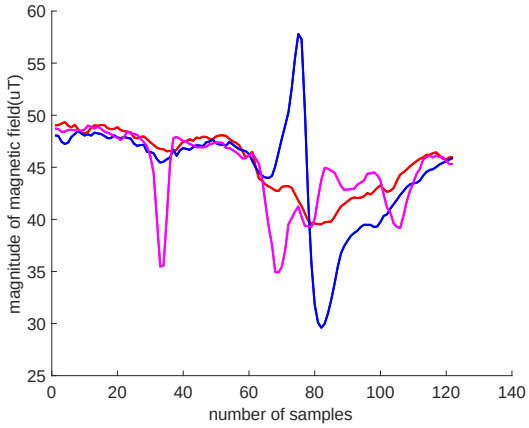


Figure 4.2: Comparing the variance of magnetic field while walking at a different distance from the wall in a hallway. Blue line: 30 cm from the wall. Red line: 60 cm from the wall. Purple line: 120 cm from the wall.

4.2.1 Magnetometer Calibration

The measured magnetic signature ideally reflects Earth's magnetic field along with magnetic fields generated by nearby ferromagnetic materials, making it feasible to utilize this uniqueness for indoor navigation. However, magnetometers may still have

significant drift caused by various distortions [17]. These distortions typically fall into two categories: hard iron or soft iron so magnetometer calibration is necessary to ensure precise measurements to eliminate these distortions.

Soft Iron Distortion

In an ideal environment, rotating the magnetometer in all possible directions results in a sphere with a radius equal to the magnitude of the magnetic field. However, materials with high magnetic permeability, such as nickel and iron, distort the magnetic field, causing the sphere to deform into an ellipsoid. Figure 4.3 illustrates this distortion in the x-y axis. One approach to mitigating soft iron distortion involves fitting the ellipsoidal magnetic data into a sphere to derive the soft iron calibration matrix $\boldsymbol{C_{sd}}$ and subsequently using $\boldsymbol{C_{sd}}$ to recover the magnetic field. Assuming the measured magnetic field affected only by soft iron distortion is $\vec{M}_{measured\_soft}$, the calibrated magnetic field $\vec{M}_{calibrated\_soft}$ can be calculated as:

$$\vec{M}_{calibrated\_soft} = \vec{M}_{measured\_soft} \times C_{sd}$$

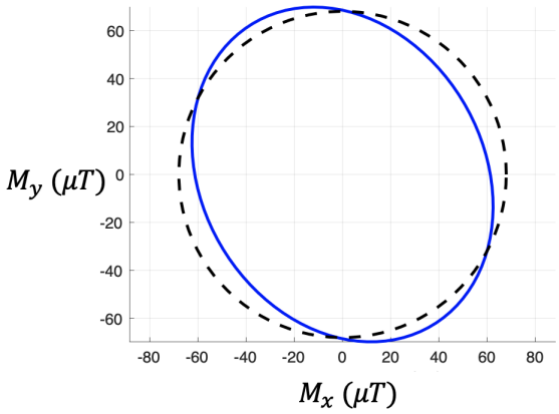


Figure 4.3: The x-y axis plot depicts soft-iron distortion in magnetometer readings. When the magnetometer is rotated along the z-axis, the black-dashed circle represents the ideal magnetic field in the x and y directions. However, soft-iron distortion makes the uncalibrated data align more closely with an elliptical shape (blue line).

Hard iron Distortion

On the other hand, hard iron distortion occurs due to permanently magnetized materials near th**e** magnetometer sensor. These materials generate their magnetic fields, introducing a constant bias to the magnetic field measured by the magnetometer. This offset remains consistent regardless of the sensor's orientation and displaces the origin of the ideal magnetic measurement sphere mentioned in the previous paragraph. Figure 4.4 provides an example of hard iron distortion in the x-y axis. Calibrating hard iron distortion is straightforward; since it creates a constant bias, we can determine the bias $\boldsymbol{b_{hd}}$ by aligning the measurement sphere with the origin.

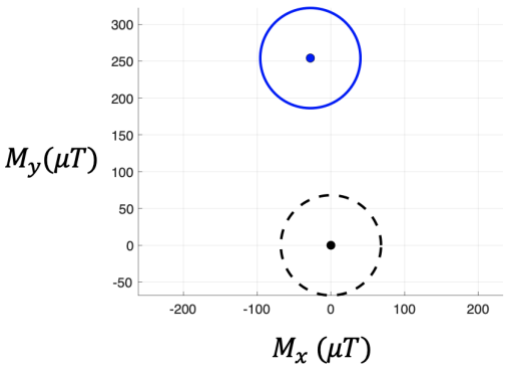


Figure 4.4: The xy-axis plot depicts hard-iron distortion in magnetometer readings. When the magnetometer is rotated along the z-axis, the ideal magnetic field orientation in the x and y directions should be centered at the origin (represented by the black dashed circle). However, the uncalibrated data may exhibit a significant bias due to hard-iron distortion, as indicated by the blue circle.

After acquiring the soft iron calibration matrix and hard iron bias, the calibrated magnetic field is represented in the following equation and used in the path-matching algorithm. Figure 4.5 shows the magnetic field before and after calibrations.

$$\vec{M}_{calibrated} = (\vec{M}_{measured} - b_{hd}) \times C_{sd}$$

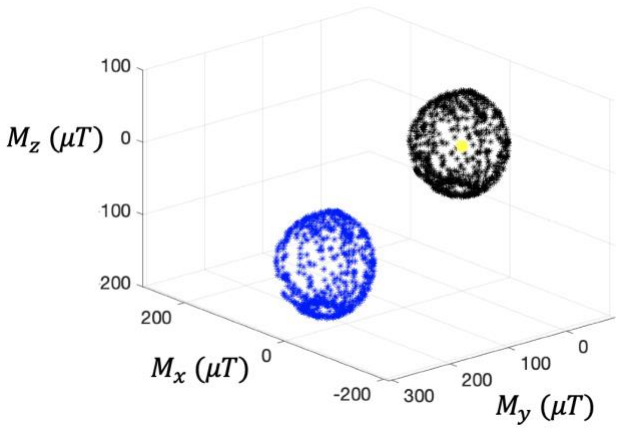


Figure 4.5: The magnetic field during calibration by rotating 360^o along all three axes. Blue: data before calibration. Black: data after calibration. The yellow dot indicates the origin point.

4.3 Cost of Magnetic Field

As mentioned in Chapter 3, our path-matching algorithm is based on sequence alignment expressed as a minimum cost route task over a properly defined graph $\mathcal{G}$. The cost matrix defined in $\mathcal{G}$ can be interpreted as the penalty of aligning corresponding elements of the sequences being matched. This penalty reflects the dissimilarity or cost of aligning two elements from the sequences, such as magnetic field readings or step/turn data. This section discusses the details of the cost of the magnetic field C_{MF} in the graph.

Let D_j as the Euclidian norm of the difference between two magnetic field vectors $\left(M_g^{in}(I_j), M_h^{in}(I_j)\right)$ and $\left(M_g^{ret}(j), M_h^{ret}(j)\right)$, where $M_g^{in}(I_j)$, $M_h^{in}(I_j)$, $M_g^{ret}(j)$, and $M_g^{ret}(j)$ are 2D magnetic fields for the corresponding I_j-*th* and *j-th* samples on the way-in and return paths, respectively. The D_j value represents difference in magnetic field

data at node (I_j, j, d_j) in the best-matched path in the graph $\mathcal{g}$, where I_j is the mapped index of the way-in for the return time instant j and d_j represents the orientation discrepancy between the mapped indices.

In the path-matching algorithm, a matched node implies that at time instant j during return, the walker is at the same location as they were at time instant I_j during way-in. Therefore, the individual likelihood of observing D_j given (I_j, j, d_j) is the correct mapping can be written as:

$$P(D_j \mid I_j), \qquad j = 1, \dots, M$$

As previously mentioned, it is assumed that when the walker is at the same location during the way-in and return, they should experience a similar magnetic field (i.e., D_j is small). On the other hand, when the magnetic difference D_j is large, there is a lower likelihood that the node represents a correct mapping, so we will assign a higher penalty to the node. Therefore, the magnetic cost of a matched pair of indices $\{I_j, j\}$ can be interpreted as the negative likelihood of magnetic difference observations. For the entire optimal matched path, if we assume that the observations are independent, the overall likelihood is represented by the product of the individual likelihoods:

$$\prod_{j=1}^{M} P(D_j \mid I_j) ,$$

where M is the number of samples in the return path. The overall magnetic cost of the path can be defined as the negative logarithm of overall likelihood:

$$-\log\left(\prod_{j=1}^{M} P(D_j \mid I_j)\right) = -\sum_{j=1}^{M} \log\left(P\left(D_j \mid I_j\right)\right) \tag{4.1}$$

To determine $P(D_j \mid I_j)$, we need to construct the histogram of the norm of differences in the magnetic field. Assuming that during the navigation, the walker does not consistently remain in the middle of the corridor due to veering behavior [52]; they can be at various distances from the wall. As described in Figure 4.2 the distance between the user and the wall affects the magnetic field along the hallway, resulting in different D_j values even when mapped into the same position along the corridor. We simplify this problem by categorizing these distances into three cases: in the middle of the hallway, close to one side of the wall, and close to the other side of the wall, as shown in Figure 4.6. In the latter two cases, we assume that the user is positioned 30cm away from the nearest wall. This distance was chosen because it falls within the average length of an upper arm (average upper arm's length ranged from 23 to 41cm for individuals aged 5 and above [53]). This measurement reflects the scenario where walkers might choose to maintain a distance from the wall equivalent to the length of their arm for safety and security reasons.

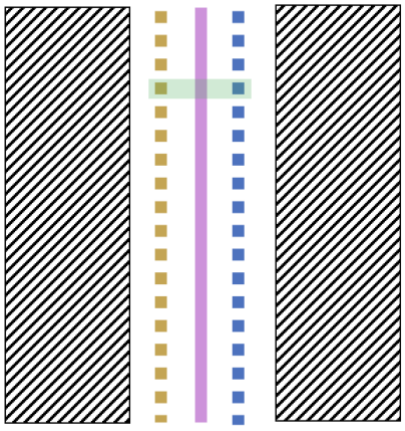
Denote $P\left(D_j \mid c_w, c_r, I_j\right)$ as the likelihood of observing D_j given c_w, c_r, and I_j (i.e., the correct mapped index for return index j is I_j), where c_w and c_r are the distance to the wall in way-in and return respectively. $P\left(D_j \mid I_j\right)$ can be calculated by:

$$P(D_j|\ I_j) = \sum_{c_w}\sum_{c_r} P(D_j \mid c_w, c_r, I_j) \times P(\ c_w, c_r|\ I_j)$$

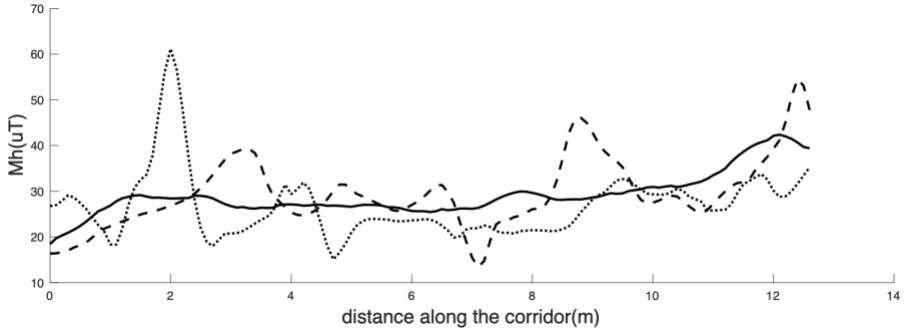
Since we don't have prior information regarding the user's walking behavior (specifically, the distance from the wall), we can consider $P(\ c_w, c_r|\ I_j)$ to be a constant, denoted as C. Therefore,

$$P(D_j|\ I_j) = C \cdot \sum_{c_w}\sum_{c_r} P(D_j \mid c_w, c_r, I_j) \tag{4.2}$$

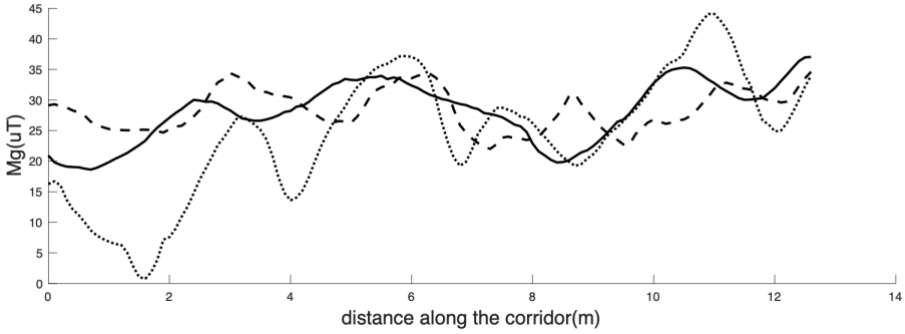
Although this method simplifies the problem into merely three distance categories, it may not fully capture the hallway's variability. However, considering the hallway width in our tested buildings typically ranges from 160 cm to 252 cm, we hope that this method can provide a reasonable approximation of the distance scenarios observed within our tested data.



(a)

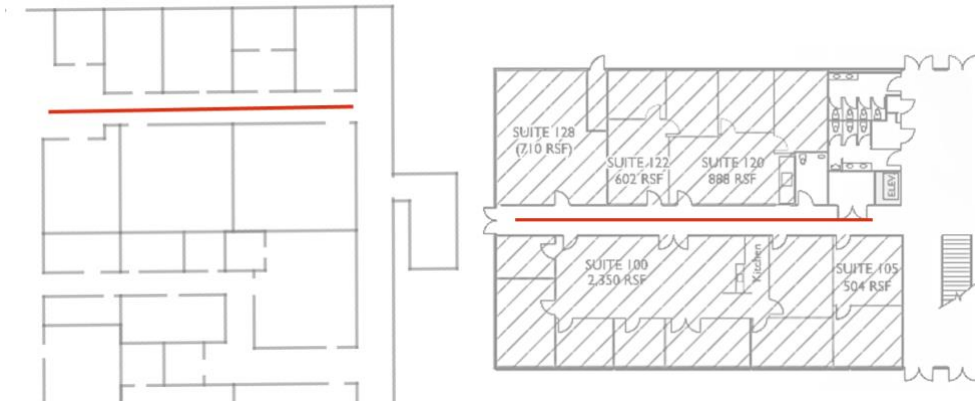


(b)



(c)

Figure 4.6 : Magnetic field measurements along the corridor. (a)The shaded areas represent the walls. The pink line represents the group of positions in the middle of the hallway, while the dashed lines indicate positions closer to the wall. The green highlighted area illustrates the same positions along the corridor but at varying distances from the wall. (b) M_h (c) M_g. For (b) and (c), the solid line is the data collected along the middle of the corridor, the dashed line is the data collected along the left side of the hallway, and the dotted line is the data collected along the right side of the hallway.



(a) (b)

Figure 4.7: Hallways (red lines) where magnetic field histograms were collected. (a) The hallway on the 2nd floor of BE building at UCSC. (b) The hallway in a local office building

We measured the norm of magnetic field differences in two hallways: one in the BE building at UCSC and the other in an office building, as depicted in Figure 4.7. The measurements were gathered by walking along each hallway in one direction (designated as the way-in path) and then retracing the path in the opposite direction (as the return path). To address variations in the walker's distance from the wall, the paths (for both way-in and return, corresponding to c_w and c_r in eq (4.2)) were also traversed at a 30 cm distance from both sides of the wall, resulting in a total of six measurements of paths.

For each path, measurements were taken at fixed intervals (0.504 m, corresponding to the average step length of our participants in the user study, which will be discussed in Chapter 7) to ensure consistent sampling. This approach allowed us to calculate D_j for a given correct mapping (j, I_j), where at the return sample index j and way-in sample index I_j, the walker was at the same location. Subsequently, the corresponding

histogram was created for $P(D_j \mid I_j)$ (after simplifying the constant C in eq (4.2) to 1) , as shown in Figure 4.8. The distribution appears to be long-tailed. Several potential distributions were considered to fit this distribution, including the Exponential Distribution, Inverse Gaussian Distribution, and Rayleigh Distribution. After examining the residuals of the fitted curves in Figure 4.8, it is evident that the Exponential Distribution (as shown in eq (4.3)) provides a superior fit, exhibiting the smallest residual value of 0.004, compared to the Inverse Gaussian Distribution (residual = 0.016) and the Rayleigh Distribution (residual = 0.041).Thus, the likelihood of observing the norm of difference in magnetic field is given by the following fitted Exponential Distribution:

$$P(D_j \mid I_j) \approx \frac{1}{\mu} e^{-\frac{D_j}{\mu}} \tag{4.3}$$

where a larger scale parameter μ indicates the more spread out the distribution.

The negative logarithm of $P(D_j \mid I_j)$ can be calculated using the following formula:

$$-\log\left(P(D_j \mid I_j)\right) \approx \frac{D_j}{\mu} + c \tag{4.4}$$

where c represents a constant. Then, the overall magnetic cost of the path in the graph (in eq (4.1)) can be reformulated as:

$$-\log\left(\prod_{j=1}^{M} P(D_j \mid I_j)\right) = -\sum_{j=1}^{M}\left(\frac{D_j}{\mu} + c\right)$$

It implies that the overall magnetic cost for the optimal matched path in $\mathcal{G}$ is the summation of the magnetic field differences D_j divided by μ, thus for each node (I_j, j, d_j) in $\mathcal{G}$, the cost of magnetic field is $\frac{D_j}{\mu}$. The estimated value of μ in the tested

data is 7.75, suggesting that in our tested buildings, the cost of magnetic field for each node is $\frac{D_j}{7.75}$. It is noted that the constant c applies to every node in our graph; thus, it can be ignored when calculating the cost.

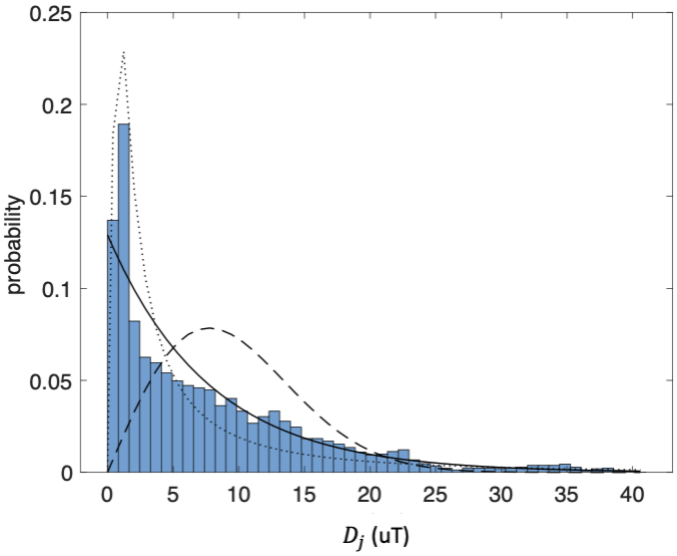


Figure 4.8: The histogram of the norm of the difference in magnetic field fitted by different PDF functions. Solid line: Exponential distribution. Dotted line: Inversed Gaussian. Dashed line: Rayleigh Distribution. (The histogram has been normalized to unit area)

A similar analysis was conducted utilizing the public dataset MINLOC [54], which includes the magnetic field collected from different corridors. However, unlike our dataset, the magnetic field at various distances from the wall was not recorded in MINLOC. Consequently, the differences in magnetic field are relatively minor. Figure 4.9 displays the corresponding histogram of magnetic field differences within the

corridors based on the MINLOC dataset. In comparison to our histogram in Figure 4.8, Figure 4.9 lacks a long tail in the histogram due to smaller differences in magnetic field. As previously mentioned, walkers may not always remain in the middle of the hallway. Therefore, incorporating measurements of magnetic differences relative to the wall provides additional insights into the characteristics of the magnetic field.

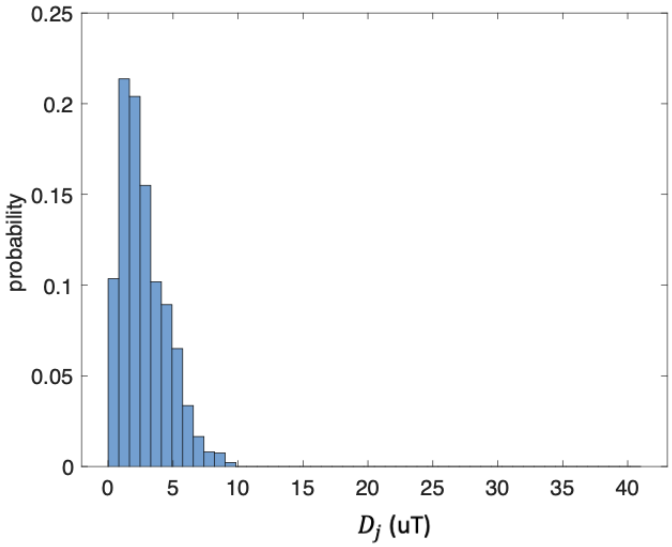


Figure 4.9: Histogram of magnetic field difference in public dataset. (The histogram has been normalized to unit area.)

4.4 Conclusion

This chapter has explored the use of indoor magnetic fields for accurate positioning and navigation. We also examined how magnetic fields can be influenced by an individual's walking behavior, specifically focusing on veering behavior observed through different measurements of the magnetic field relative to various distances from the walls.

Calibrating magnetometers to correct distortions caused by nearby metals is critical for achieving accurate measurements. Two main types of distortions, hard iron and soft iron, were discussed along with methods to calibrate magnetometers to mitigate these distortions.

Additionally, we also explored how to leverage differences in magnetic fields between mapped locations (referred as the "cost of magnetic field") in the path-matching algorithm. This involved studying the likelihood of the observed norm of difference in the magnetic field, enhancing our algorithm's capability to manage variations in magnetic field data for precise positioning purposes.

Chapter 5

Path-matching Algorithm: Experiment with WeAllWalk Dataset and On-Site test

In this chapter, we present the experimental results of the path-matching algorithm. The experiments are divided into two parts. In the first part, we simulated assisted return situations using the WeAllWalk dataset to compare different odometry systems that use inertial data ($k \cdot 90^o$ or $k \cdot 45^o$ turn detector w/wo step information, where k is an integer). The odometry system exhibiting a smaller path-matching error was selected for integration with magnetic field information in the second part of the on-site test conducted with the SafeReturn app. This test was conducted in the E2 building at UCSC. Additionally, the interface used for evaluation is presented in this chapter.

5.1 Comparing Path Odometry Algorithms for Assisted Return – WeAllWalk Experiments

The WeAllWalk dataset [22] contains inertial sensor data, which was collected from six visually impaired participants while they walked through several pre-defined trajectories using a walking cane or a dog guide. Our path-matching algorithm

leverages both steps/turns and magnetic field data to map the current position during the return to a position collected during the way-in phase. While steps/turns information for the WeAllWalk dataset can be obtained by processing the inertial data using step and turn detectors developed by other PhD researchers in our lab [46][12], the magnetic field data in the WeAllWalk dataset was not calibrated. Therefore, we only utilize the steps/turns data when evaluating our algorithm in this part of the experiments.

We simulated an assisted return situation where a certain walker traversed the entire path first (as the way-in path), followed by another (or the same) walker traversing the same path while incrementally matching their path with that of the first walker (as the return path). We considered pairs of traversals for each path, either by two different walkers or by the same walker using different mobility tools (cane or dog guide). Figure 5.1 illustrates an example of these paths. There were 162 such traversal pairs on which our path-matching algorithm was tested. For each of them, the optimal alignment matching was computed incrementally for each time instant during return. It is noted that the off-route scenarios are not included in this experiment but are included in the next section for the on-site tests (section 4.3).

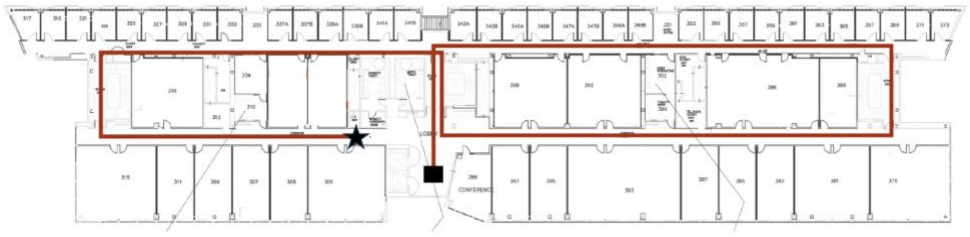


Figure 5.1: One of the six paths from the WeAllWalk dataset. The path begins at the square and ends at the star [22].

Odometry System	k · 90°turns	k · 45° turns	k · 90° turns + steps	k · 45° turns + steps
E_{PM}	5.28	6.43	**4.17**	4.50

Table 5.1: Path-matching error E_{PM} (in seconds) measured using different path odometry systems for the WeAllWalk experiment. The integer ' k ' represents the system's capability to detect different turn angles.

As mentioned in the previous chapter (section 3.5), we use an approximate location (serves as reference data) to calculate the path-matching error E_{PM} and

Table 5.1 shows the recorded average errors using different turns/steps odometry, with turns computed either at a multiple of 45° or 90°. (Note that 13% of all turns in WeAllWalk are ±45° turns.). It is seen from

Table 5.1 that the lowest average error was obtained using both the turns (multiple of 90^oturns) and steps representation in the definition of graph costs.

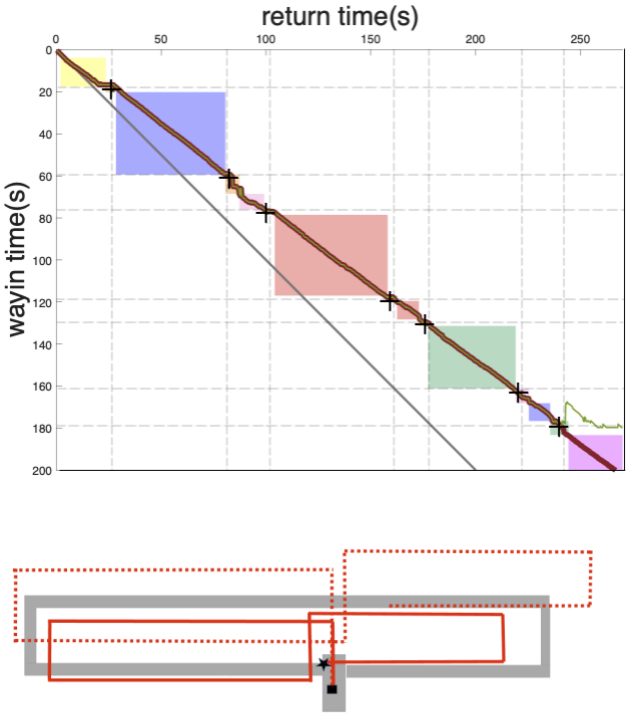


Figure 5.2: The best matching sequence for two walkers, one using a dog guide and the other using a long cane. The colored rectangles represent the entry and exit time of each "segment," as marked in WeAllWalk. The '+' signs represent 90^o turns. Red line: $k \cdot 90^o$+ steps (mean error: 0.8 s); Gray line: baseline (mean error: 27.1 s); Green line: $k \cdot 45^o$+ steps (mean error: 0.8 s). The horizontal and vertical lines show 90^o (dashed) turns detected during way-in and return, respectively. Bottom: the reconstructed paths by the $k \cdot 90^o$+ steps odometry (without using the path-matching algorithm to match the return samples to way-in samples) plotted on the building map. Solid line: way-in path. Dotted line: return path. The way-in path starts from a square and ends in a star.

To provide some insight into the results, examples of path matches for pairs of walkers over the same path are shown in Figure 5.2 and Figure 5.3 (top), while the reconstructed paths are shown against a map of the building (in the bottom of the figures and the reconstructed path was plotted assuming a stride length of 0.567 m.). In the path-matching plots, the vertical and horizontal axes indicate time instants during way-in and return, respectively. The colored rectangles represent contiguous segments in the

path. Specifically, the vertical coordinates of each rectangle's top and bottom edges correspond to the times when the walker entered and exited the rectangle during the way-in (as recorded in WeAllWalk) and similarly for the return path. The diagonal line joining each rectangle's top left and bottom right corners (not shown in the figures) represents the set of correct nodes $\{(\hat{l}_j, j)\}$ using the interpolation approximation described above. The '+' and '$*$' signs in the plots represent time instants (for both way-in and return) in which a 90^o or 45^o turn was marked in WeAllWalk. For each path-matching algorithm displayed in the figures, a line represents the set of nodes selected by the algorithm. Lines close to each rectangle's diagonals denote satisfactory path matches. A "baseline" match of all time instants in way-in with corresponding time instants in return $\{(i, j)\}$ is also shown, which assumes that the two participants walked at the same speed.

Figure 5.2 compares the path-matching using $k \cdot 90^o+$ steps (red line) and $k \cdot 45^o+$ steps (green line) odometry. In the path-matching plot(top), both methods show similar results up to return time t = 240s. However, after that point, the $k \cdot 45^o+$ steps deviates from the rectangle's diagonals. It can be observed that the $k \cdot 90^o+$ steps produces a slightly better result than the $k \cdot 45^o+$ steps, while both are substantially better than the baseline.

Figure 5.3 shows an example using the $k \cdot 45^o+$ steps algorithm. In this case, the path had one 45^o+ turn at the beginning, followed by two 90^o turns. The graph shows horizontal and vertical lines corresponding to the time when a 45^o (dotted) or 90^o(dashed) turn was detected during way-in or return, respectively. The algorithm

correctly detected the 45^o turn, but the second 90^o turn was mistakenly detected as a sequence of two 45^o turns during return. The path-matching algorithm was able to manage this situation correctly. However, some "jitter" is noticeable (see segment marked in green), which is a consequence of the fact that dynamic programming was implemented incrementally (rather than just at the end of the return path).

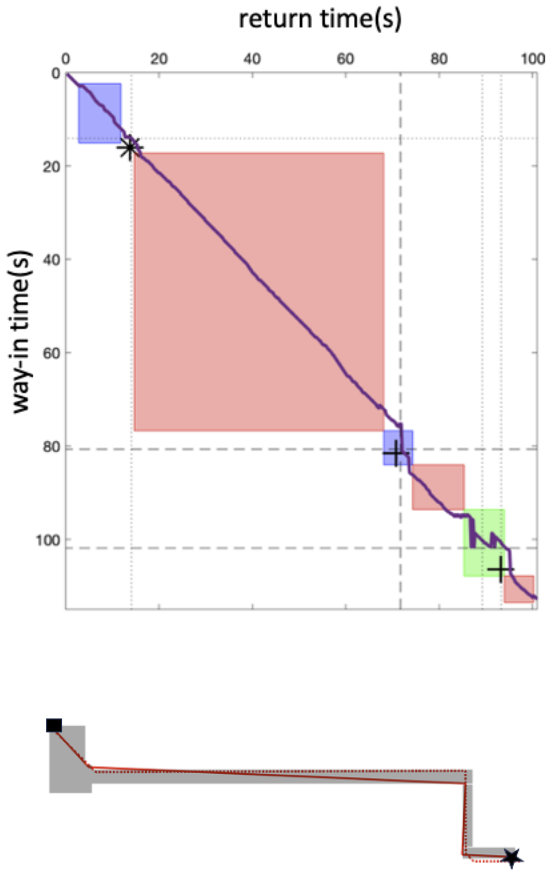


Figure 5.3: The top plot represents the best matching sequence with the $k \cdot 45^o$+ steps algorithm for two walkers, both using a long cane. The colored rectangles represent the entry and exit time of each "segment," as marked in WeAllWalk. The '+' sign represents 90^o turns, while the '*' sign represents a 45^o turn. Purple line: $k \cdot 45^o$+steps (mean error: 2.72 s). The horizontal and vertical lines show the 45^o (dotted), and 90^o (dashed) turns detected during way-in and return, respectively. Bottom: The

reconstructed paths are overimposed on the building map. Solid line: way-in path; Dotted line: return path.

5.2 SafeReturn App - User Interface for Evaluation

An IOS App was built to evaluate the path-matching algorithms in real-world environments. This application was tested extensively on an iPhone XR. The application's user interface during the test process is shown in Figure 5.4. There are four application views, and the arrows between the views indicate that users can navigate directly between them. Figure 5.4(a) is the initial view for users to decide whether starting the path-matching task or calibrating the magnetometers. Figure 5.4(b) is the calibration view for users to calibrate the magnetometers by rotating the smartphone alone on the three axes. This will produce a plotted solid circle representing the completion of the calibration (Note that it is recommended to perform calibration before starting a way-in path). The primary operation of path-matching is performed in Figure 5.4(c). The trajectory and the mapped samples between way-in and return paths are plotted in the view. Figure 5.4(d) is the setup for parameters, and a detailed description of the parameters is provided in the Appendix.

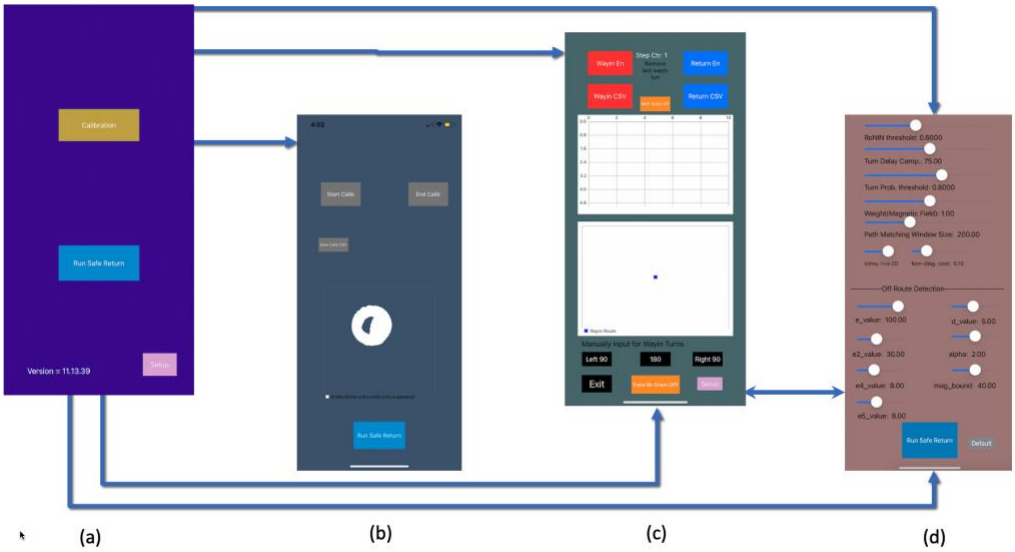


Figure 5.4: Design of the user interface. (a): entry view (b): calibration view (c): the main view for path-matching (d) parameter setup view.

5.3 On-Site Test

The on-site test was conducted in the E2 building at UCSC to evaluate the system's capability to provide path recovery guidance in situations where the walker deviates from the correct path. Figure 5.5 illustrates an example of successful path recovery, and its corresponding path-matching graph is shown in Figure 5.6. On the return journey, the walker was asked to miss the first 90^o turn to simulate an off-route situation. In the path-matching plot, the green line indicates the best path-matching sequence determined by the algorithm, which is only available at the end of the return trip. On the other hand, the black line represents the matching sequence calculated on the fly, which is also the data used to provide real-time guidance.

Figure 1.6 shows a significant overlap between the black and green lines, indicating that real-time guidance closely approximates the optimal path calculated after collecting all return data. This suggests that real-time guidance reflects optimal matching. In the sequence, points with non-zero orientation discrepancy ($d \neq 0$) or off/reversed-route status ($d = 4 \; or \; d = 2$) are highlighted in different colors. The yellow dots at the return sample #120 indicate that the system detected an off-route status and prompted guidance for the walker to make a U-turn. The walker followed the instruction, and then a reversed-route status was detected at return sample #240 (brown dots in the figure), which is expected because the walker was on the correct path but facing the opposite direction. The system provided further instructions to prompt another U-turn, eventually guiding the walker back onto the correct path. While other false-positive reversed-route points occurred between return samples #131 and #183 (brown dots), these were only generated in the best path-matching sequence at the end of the return trip. Therefore, they weren't detected during the real-time navigation.

The system successfully mapped samples on the return path to those on the way-in path and provided path recovery to the walker. The corresponding trajectory of the path and the path-matching plots on the app are also shown in Figure 5.7.

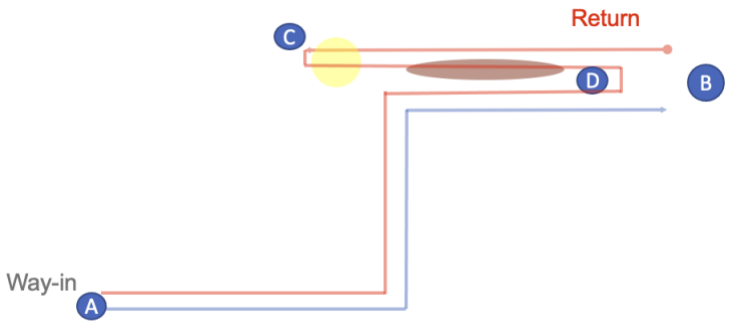


Figure 5.5: Illustration of trajectory with successful path recovery guidance generated by the system. The way-in path is from A to B, and the return path is from B to A. The off-route and reversed-route segments are highlighted by the yellow and brown markers.

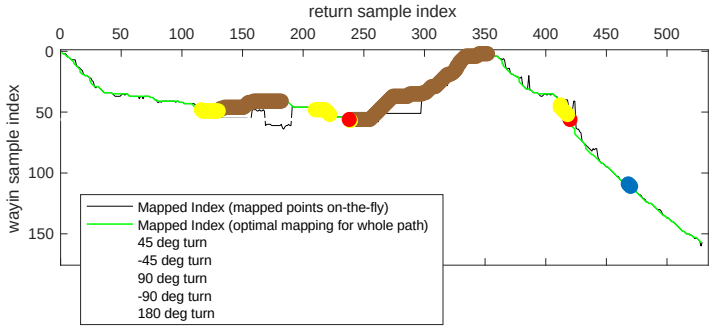


Figure 5.6: A representation of the best path-matching sequence. Horizontal lines: 90° (red) and -90° (blue) turns detected during way-in. Vertical lines: 180° (brown), 90° (red) and -90° turns detected during return. Green line: The best path-matching sequence selected by the algorithm at the end of the return. Colored cluster: Points with non-zero orientation discrepancy (layer $d \neq 0$) or off-/reversed-route status in the best path-matching sequence. Black line: Path-matching sequence computed from return data up to the real-time return sample index.

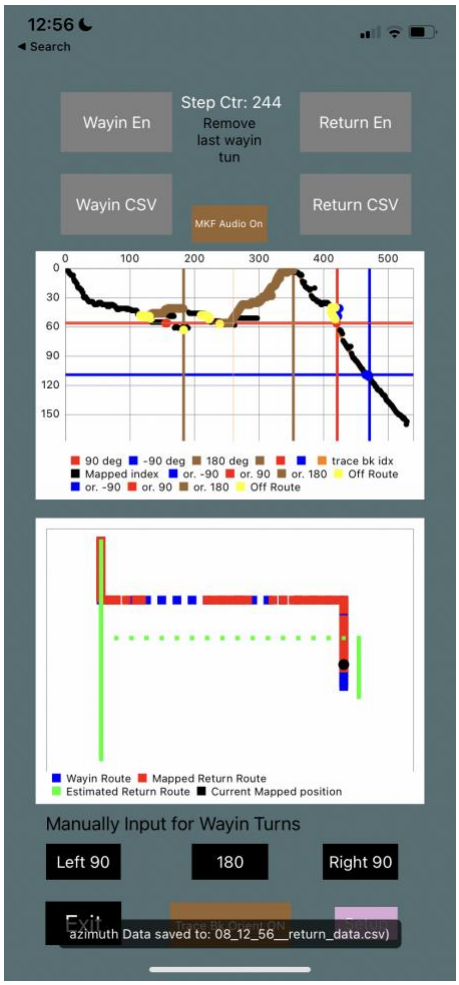


Figure 5.7: The view in the SafeReturn app. Top plot: Best path-matching sequence. Bottom: Trajectory of the path.

The previous example shows a scenario where the system provided accurate real-time guidance and identified instances when the user went off-route. However, the effectiveness of detecting off-route situations has limitations when there is dynamic magnetic interference from sources like the running elevators in our tested building, as highlighted in another scenario depicted in Figure 5.8 for trajectory and Figure 5.9 for the path-matching graph.

Similar to the initial example, during the return route, the walker failed to execute a 90^o turn and continued straight, leading to an off-route situation (highlighted in yellow in Figure 5.8). Despite the ability of the graph to identify the walker's off-route status upon collecting all return data at the end of the trial (depicted by the green line in Figure 5.9, representing the optimal graph path), it failed to offer real-time guidance (illustrated by the black line in Figure 5.9) when the walker deviated from the route, as the green line (with the yellow dots) and the black line start to diverge from same index #420. This discrepancy is attributed to the similarity of magnetic field signatures across different locations, resulting in incorrect location identification within the system. When two distinct locations exhibit similar magnetic field signatures or when the magnetic field is temporarily affected by large metallic objects, it can result in incorrect mapping. Another drawback of solely relying on the path-matching graph for localization is that it always assigns a mapped location to the way-in for every incoming return sample. However, this mapping may not always be correct if the walker deviates from the correct path. To overcome these challenges, we updated the algorithm to incorporate the concept of reliable matching. This enhancement will be further elaborated on in the upcoming chapter.

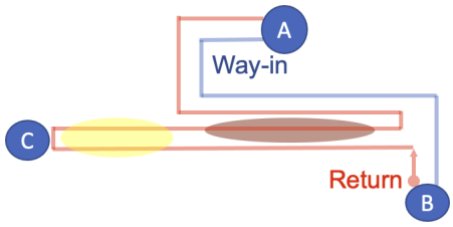


Figure 5.8: An illustration of a trajectory where off-route deviations were not successfully identified in real-time. The path originates from point A to point B (way-in path), followed by the return path from point B to point A. Off-route and reversed-route segments are highlighted by the yellow and brown markers, respectively.

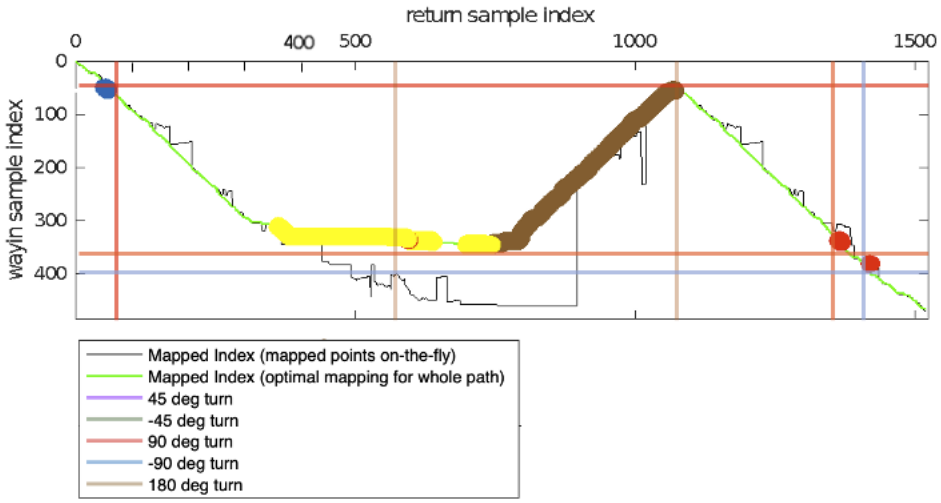


Figure 5.9: A representation of the best path-matching sequence where off-route deviations were not successfully identified in real-time. Horizontal lines: 90º (red) and -90º (blue) turns detected during way-in. Vertical lines: 180º (brown), 90º (red) and -90º (blue) turns detected during return. Green line: The best path-matching sequence the algorithm selects at the end of the return. Colored cluster: Points with non-zero orientation discrepancy (layer $d \neq 0$) or off-/reversed-route status in the best path-matching sequence. Black line: Path-matching sequence computed from return data up to the real-time return sample index.

5.4 Conclusion

In this chapter, we conducted simulations of assisted return using the WeAllWalk dataset to compare the odometry systems based on robust two-stage turn detection and step counting. The analysis revealed that the $k \cdot 90^o+$ steps algorithm performs better than other methods. Furthermore, we developed an iOS app for our system and performed on-site testing to assess its performance. Results indicate that the system effectively maps the user's position and offers path recovery assistance when the magnetic field is stable. However, certain limitations arise in scenarios where the magnetic signature is affected by other sources and becomes unreliable. Therefore, in the following chapter, we propose a hybrid matching approach to enhance the system's performance and achieve more robust results.

Chapter 6

Enhanced Path-Matching Algorithm:

Hybrid Matching with Last Reliable Position

In the basic path-matching algorithm (Chapter 3), we only utilized the path-matching graph $\mathcal{g}$ to generate the mapped position during the return phase. However, as discussed in previous chapters, relying solely on the graph $\mathcal{g}$ presents several challenges. For instance, temporary disturbances in the magnetic field can lead to incorrect matching. The accuracy of the matching process is significantly influenced by the stability of the magnetic field. Consequently, this chapter introduces the enhanced path-matching algorithm: hybrid matching, which incorporates the concept of the "last reliable position (LRP)" to enhance the robustness of the system. Two methods for determining LRP are introduced, and the algorithm is evaluated using datasets from various buildings at UCSC.

6.1 Last Reliable Position (LRP)

The idea of identifying “last reliable position (LRP)” or "last known position" is commonly employed in some navigation systems. For instance, in GPS navigation,

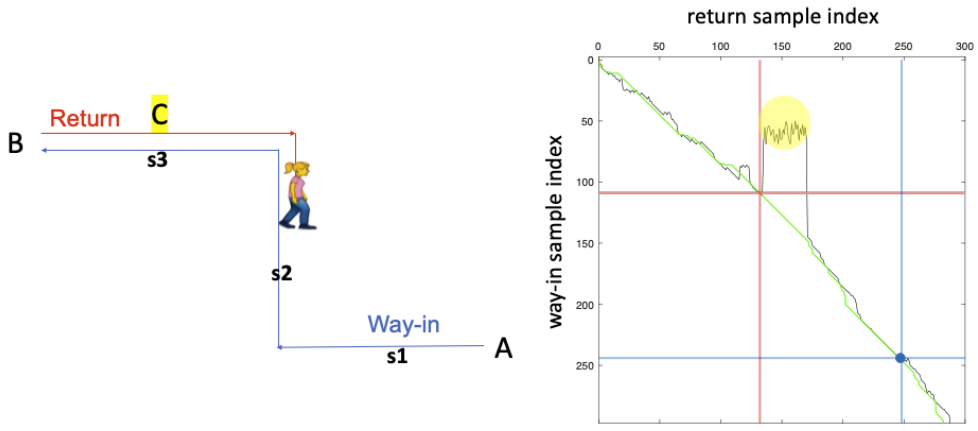
Receiver Autonomous Integrity Monitoring (RAIM) evaluates its integrity by analyzing pseudo-ranges between satellites and the receiver. If anomalies or issues are detected, RAIM takes corrective action by excluding signals from problematic satellites or indicating the current GPS positioning cannot be trusted. When such situations arise, alternative solutions, such as integrating with other sensors, can be implemented, and the current position can be calculated based on the last known position [55]. This concept is not limited to GPS alone but is also applied in Wi-Fi-based [56] and iBeacon-based [57] indoor positioning systems. In these systems, nodes exhibiting anomalies must be eliminated before calculating the last known position, particularly before providing input to the Pedestrian Dead Reckoning (PDR) system.

When backtracking the walker's position in our application, leveraging the concept of the Last Responsible Position (LRP) offers two distinct advantages: **preventing contradictory guidance** and **identifying deviations from the correct path**.

Preventing Contradictory Guidance:

When providing real-time navigation for visually impaired individuals, excluding unreliable localization information is particularly crucial to prevent contradictory guidance, which can confuse the users. Figure 6.1 illustrates such a scenario: the entire return path is S3 (straight segment) → "right turn" → S2 (straight segment) → "left turn" → S1 (straight segment). At the first junction, the walker correctly makes a right turn towards the next **left turn**. However, due to a temporary disturbance in the magnetic field, the graph places the walker back into the previous segment (highlighted

in yellow in Figure 6.1 (b)). As a result, the system announces inaccurate guidance, such as "At the upcoming junction, make a **right turn**." As more data is acquired (e.g., the walker takes a few more steps), the path-matching graph can accurately locate the walker, providing the correct guidance: "At the upcoming junction, make a **left turn**." The inconsistency between these guidance messages (left turn vs. right turn) underscores the importance of determining the reliability of positional information to avoid providing conflicting information to the user.



(a) (b)

Figure 6.1: An illustration of inconsistent guidance without adopting LRP. (a) The trajectory of the walker: The way-in path starts from point A to B, and the return path starts from point B. (b) The x-y axis of the path-matching graph of the whole sequence of paths. Red lines and blue lines indicate 90^o and -90^o turns, respectively. The walker makes a correct 90^o turn, but the graph initially misplaces the walker into a prior segment for a short period of time and asks the walker to make a 90^o turn again (highlighted in yellow in (b) and point C in (a)). Subsequently, it locates the walker to the correct position and instructs the walker to make a -90^o turn. However, this inconsistency may result in confusion.

Identifying Deviations from the Correct Path:

The original path-matching algorithm consistently assigns a mapped location to the way-in for every incoming return sample. However, this mapping may become inaccurate if the walker deviates from the correct path. Figure 6.2 (a) shows an example of this scenario. The walker missed the 2nd turn junction and continued straight ahead. Still, the path-matching algorithm incorrectly assigns a mapped location to the way-in, potentially near the destination (point C in Figure 6.2 (a) and the highlighted area in Figure 6.2 (b)), misleadingly informing the walker that they are "approaching the destination."

Although we may utilize the off-route layer ($d \neq 0$) in the graph $\mathcal{G}$ to identify deviations from the intended path, as discussed in Chapter 5.3, the reliability of this information can be compromised by external factors affecting the magnetic field (e.g., nearby running elevators), making the results unreliable.

In contrast, in a hypothetical scenario where the last reliable position (LRP) can be identified, as shown in Figure 6.2 (c), if the walker misses a turn and continues walking, we can then project their position from the LRP (pink dot). The projected positions from LRP can help us to determine whether they have deviated significantly from the expected turn junction. This approach allows us to identify deviations from the correct path more accurately.

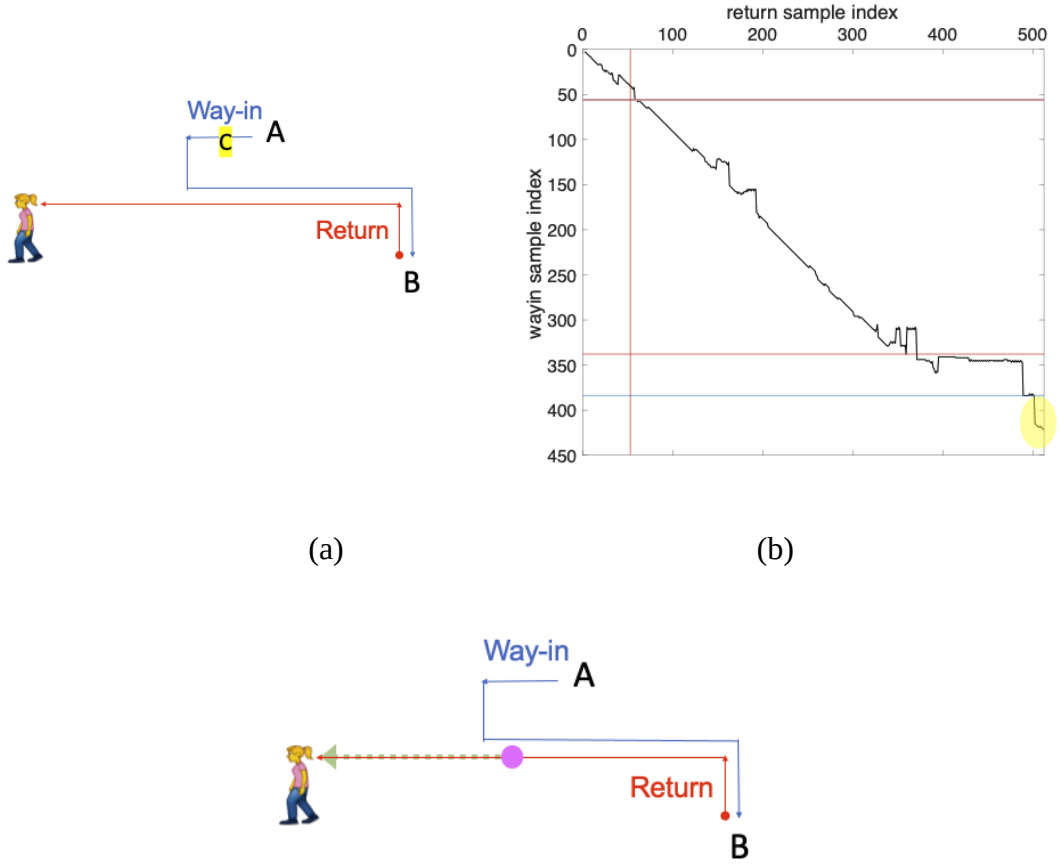


Figure 6.2: (a) and (b) illustrate inconsistent mapping without adopting LRP. (a) is the trajectory of the walker: The way-in path starts from point A to B, and the return path starts from point B. (b) is the x-y axis of the path-matching graph of the whole sequence. Red lines and blue lines indicate 90^o and -90^o turns, respectively. The walker misses the turn, but the graph misplaces the walker into a location close to its destination (highlighted in yellow in (b) and point C in (a)). (c) shows the projections from LRP (pink dot). The green dashed line indicates the walker's path after LRP was found.

6.2 Notation

Before delving into the details of how LRP works in our design, the notation used throughout this chapter is introduced here to ensure clarity and consistency in our discussions.

$S(j)$: The whole sequence for mapped indices at time instant *j*, i.e., the minimum path traced back from the $j-th$ column of the cost matrix the green line in Figure 6.3(b)). At the end of the return path, the optimal mapped sequence is $S(M)$ where M is the number of samples of the return sequence.

(i_j, j, d_j) : The triplet of a mapping at time instant *j*, where i_j is the corresponding mapped index of the way-in and the layer d_j represents its orientation discrepancy. It is noted that (i_j, j) is the last element of the mapped indices in the $S(j)$ and connecting (i_j, j) for all $0 < j \leq M$, creating the black line in Figure 6.3 (b))

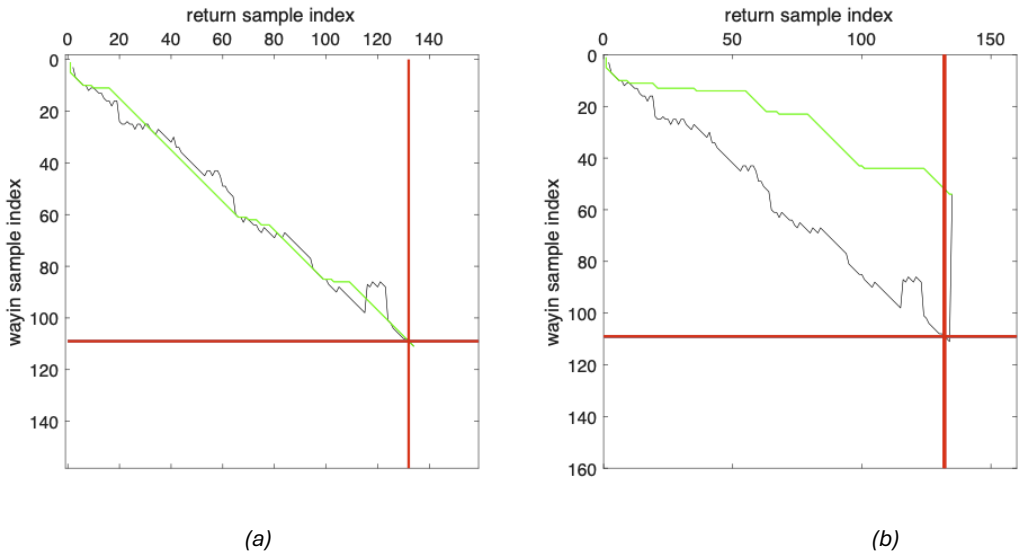
M_h^{in}, M_g^{in} : the recorded (in reverse time order) 2-dimensional magnetic field during way-in.

M_h^{ret}, M_g^{ret} : the recorded 2-dimensional magnetic field during return.

6.3 LRP in the Path-Matching Graph

When mapped positions remain consistent and reliable over time, they should not contradict each other. For example, assuming the user walks at a constant speed in both the way-in and return phase, during the return phase, if the user walks along the same path as the way-in path, the mapped indices should be gradually increasing. This scenario is illustrated in the path-matching graph in Figure 1.3 (a). As the user progresses along the return path (x-axis), the mapped indices gradually increase (y-axis).

However, this consistency is not always guaranteed in real-time navigation scenarios. In real-time backtracking navigation, samples from the return path are aligned to the original way-in, with the optimal matching computed at each time step denoted as $S(J)$. As the user progresses along the return path, a new optimal matching $S(J+1)$ is generated. This updated matching may not always include the previous one. As depicted in Figure 6.3 (b), during the return phase, the mapped index of the way-in path shifted significantly from around #110 to approximately #50 after the user made a 90^o turn near return sample index #130. To assess the reliability of the mapped indices, we propose two methodologies to assess whether the current matching is reliable: one is based on linear fitting of the matched path in the graph $\mathcal{G}$ and the other is based on machine learning. They are described in the following sections.



(a) *(b)*

Figure 6.3: An illustration of different optimal matching (green line) in consecutive time instants in the x-y axis of the path-matching graph. Green line: the optimal matching sequence (i.e., $S(j)$) from the most recent time instant. A. Black line: the best match indices for every time instant (i.e., (i_j, j)). Red lines and blue lines indicate 90^o turns, respectively. (a) The graph at time instant *J*. (b) The graph at time instant *J+1*.

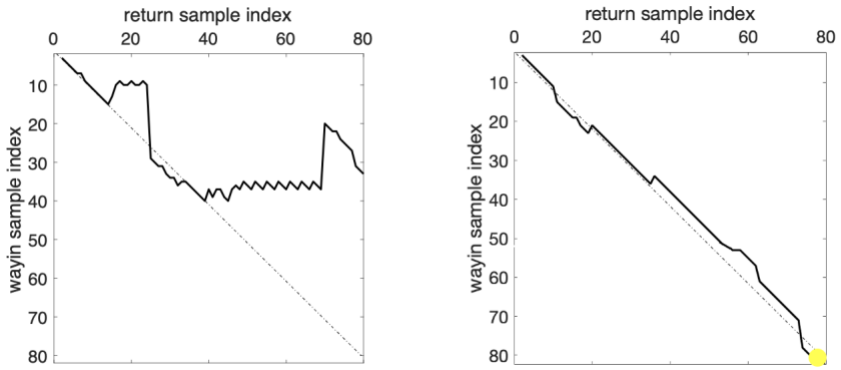
6.3.1 Linearly Defined LRP

As mentioned earlier, mapped positions should remain consistent over time. In this method, we determine LRP by examining whether the mapped position at the current time instant contradicts earlier calculated positions. We use local properties of the current minimum cost graph path to decide, at the current time J, whether the match (i_J, J, d_J) can be considered "reliable," meaning that it is likely to be preserved even after later observations are recorded and the mapped points are toward the same directions (i.e., $d_J = 0$). As illustrated in Figure 6.4 (a) and (b), the mapped positions exhibit fluctuations over time in scenario (a), indicating potentially unreliable matchings. Conversely, in scenario (b), the samples are matched smoothly over time, suggesting that some can be considered reliable.

In practice, we look at the last *N* samples of the minimum cost path in the graph ending at (i_J, J, d_J).

If this path segment aligns well with a line with a unitary slop (indicating that the residual of the fitting falls below a predefined threshold) and also it exhibits zero orientation differences ($d_J = 0$, i.e., no orientation discrepancy between the mapped way-in and return indices), then we can identify the latest mapped time instant as the last reliable mapped point. Its corresponding mapped position is designated as LRP. The set of LRPs determined by this method for the entire return path is denoted as LRP_{local}.

As time progresses, we continuously evaluate the past *N* samples to determine if a new LRP has emerged. Consequently, in Figure 6.4 (b), multiple LRPs may be identified over time, but our focus is solely on the most recent LRP for navigation guidance and positioning purposes. It is noted that we chose N=21 in our system. The decision is based on our initial experiments, where we found that setting *N* to 21 provided a sufficient samples of magnetic field samples to establish a reliable mapping.



(a) (b)

Figure 6.4: Two examples of mappings (with layer d=0 in both plots) from different experiments. Black line: the best match indices (i_J, J) for every time instant. Dashed line: a line with a unitary slope. (a) The mapped indices cannot be fitted into a unity slope, leading to an undetermined LRP. (b)The LRP is successfully determined and indicated by a yellow solid circle.

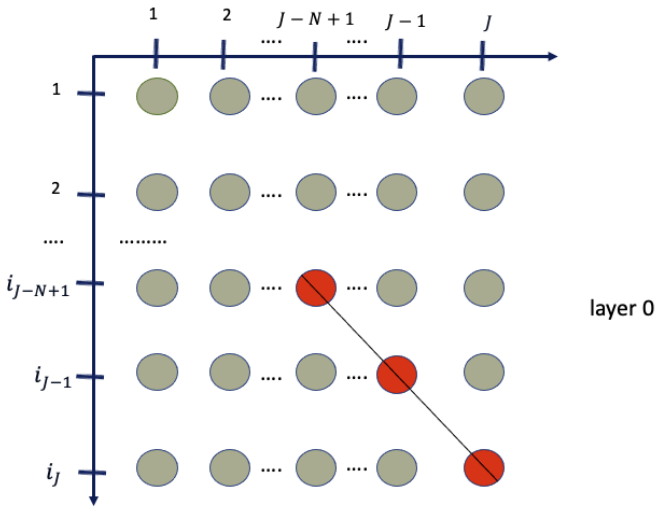


Figure 6.5: An illustration of determining the LRP in the path-matching graph $\mathcal{g}$. The X-axis and Y-axis are mapped indices during return and way-in, respectively. The gray solid circles are all the nodes $(\mathrm{i}, \mathrm{j}, 0)$ where it is assumed that the orientation discrepancy $d = 0$ to simplify the graph. The red nodes are the chosen nodes $(i_J, J, 0)$ with a minimum cost at each incoming time instant during return. The black line implies that the walker is progressively moving forward, so the node on the lowest right corner is considered as the last reliable mapping, and its corresponding mapped position is LRP.

However, this method merely considers the last mapped indices (i_j, j) from $j = J - N + 1\ to\ j = J$ and their orientation differences d_J, potentially overlooking features extracted from the magnetic field and the knowledge of sequences $S(J)$, i.e., the green lines in Figure 6.3. Therefore, we introduce another method to incorporate this information in identifying the last reliable position.

6.3.2 LRP Determined through Machine Learning

The uniqueness of the magnetic field at a certain location (along with steps/turns information) is used in our algorithm to match return samples to their corresponding way-in sequences. Theoretically, correctly matched samples should exhibit similar magnetic field patterns, indicating reliable mapped positions. However, temporary disturbances in the magnetic field can lead to incorrect matching (as shown in the prior example in Figure 6.2(b)). To address this, an alternative approach was proposed. We incorporate both the path in the segment (the last *N* samples of the minimum cost path) and magnetic field information and utilize a neural network to determine the reliability of the match, rather than relying solely on linear fitting and residual thresholding of the path segment. Here's our approach:

For every time instant during return, we have the following information: the mapped indices (i_j, j, d_j), and two-dimensional magnetic field corresponding to both way-in and return sequences, $(M_h^{in}(i_j), M_g^{in}(i_j))$ and $(M_h^{ret}(j), M_g^{ret}(j))$. We focus on the last *N* samples of this information, where *j* ranges from *J-N+1* to *J*, ensuring a consistent

length of *N* for each set. Additionally, we have the optimal sequence of mapped indices, denoted as $S(J)$, obtained by tracing the minimum path back from the *J-th* column of the cost matrix**.** We extract the last *N* paired elements of $S(J)$, resulting in a size of *2N*. Notably, (i_J, J) represents only the last element of $S(J)$. Combining all the information (size of *Nx9*), we can employ a neural network to identify the LRP.

To train the neural network for this task, we must first establish the ground truth of the last reliable positions. The optimal path $S(M)$ generated at the end of the return path was utilized to define the ground truth, because this is the most reliable match we can find. While this assumption may not always hold true, it represents the best-calculated result achievable after acquiring all of the return data.

We filter out points in the path (computed at the end of mapped sequence $S(j)$) using these empirical criteria to ensure their reliability:

1. A reliable point (i_J, J) should not deviate significantly from the optimal path $S(M)$. Specifically, the shortest distance between $S(M)$ and (i_J, J) should be smaller than a predefined threshold (dist((i_J, J), $S(M)$) <= threshold).
2. A reliable point (i_J, J) should be mapped to the same straight segment of the trajectory determined by the optimal path $S(M)$, ensuring the coherence of the mapping.

3. The consecutive points in a horizontal line in the path-matching graph are considered reliable matches because they suggest that the user remains at the

same position during the return. However, this contradicts the nature of the sequences of time indices, which are defined based on each detected step. Since each detected step should ideally represent a change in the user's position (assuming the detected step is not taken in the same location), the user's position should also change between the time indices. Therefore, $(i_J - i_{J-1}) = 0$ or $(i_{J+1} - i_J) = 0$, which means a horizontal line in the graph, cannot be considered as a reliable match.

To better illustrate the third criterion, two plots are shown in parallel in Figure 6.6, with labeled reliable matches shown in purple (unreliable matches are highlighted in yellow) for the same dataset. The walker deviated from the correct path between return time index #210 to 290 (points in the circle on both figures). In the right figure, the third criterion was not applied, resulting in points within the circle area being incorrectly labeled as reliable. After applying the third criterion, those points are indicated as unreliable.

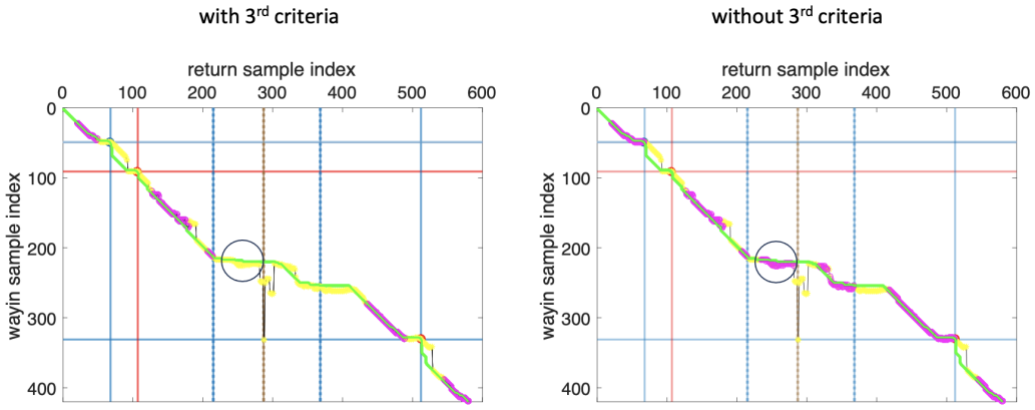


Figure 6.6: An example of a graph indicating the importance of taking the 3rd criterion while defining the LRP. See the caption of Figure 6.3. Purple markers: labeled reliable matches. Yellow markers: unreliable matches. Right: not taking the 3rd criterion when

determining LRP (points in the circle are incorrectly labeled as reliable). Left: adopting all three criteria.

Once all three criteria are satisfied, a point (i_J, J) computed at the end of mapped sequence $S(j)$, is considered reliable, and we utilize it as ground truth during the neural network training. It is noted that if the walker deviates from the correct path (for example, the walker missed a turn), the corresponding mapped positions are not considered reliable. The set of last reliable positions in the ground truth for the entire return path is denoted as LRP_{gt}.

Five different types of networks were tested in this study to determine the last reliable positions, the output of a generic model is called LRP_{NN}: fully-connected network (FCN), long short-term memory (LSTM), 1D convolutional network, graph neural network (GCN) [58], and graph attention network (GAT) [59]. Each network has a similar number of parameters, including one input layer, one output layer, and one hidden layer, totaling around 7K parameters. In the case of FCN, the input data with a size of $N \times 9$ were flattened. For GAT and GCN, we utilized a graph representation of the data, dividing it into two groups of nodes: way-in nodes and return nodes. Each group contains 5 features, including two-dimensional magnetic field data (M_h, M_g), orientation differences (layer *d*), mapped indices on-the-fly (i_J, J) (i.e., the black line in the graph), and the optimal mapped indices sequences $S(j)$ (i.e., the green line in the graph). It's important to highlight that the orientation differences are the same for both groups.

6.3.3 Projected Positions Based on LRP

When a reliable match is detected, a projected return sequence is initiated and continuously adjusted until a new reliable match is identified. This sequence originates from the last reliable match, aligning its initial direction with the walker's current orientation determined by its return path. Figure 6.7 illustrates the user’s position based on LRP over a period during the return path.

Utilizing this projected sequence, the system generates guidance notifications based on the walker's location. Upon detecting a new reliable match, the projected sequence is re-initialized at that point. Visual representations of reliable matches and projected paths are provided in Figure 6.8. In these figures, the "way-in route" is depicted with a prominent purple line, with segment lengths determined by multiplying the step count by the average step length generated by the WayFinding app, which will be further discussed in the next section. For this reason, these segments may not perfectly align with the corridors depicted in the underlying floor plan. Nonetheless, this discrepancy doesn't pose an issue, as the primary objective of the system is to accurately match the walker's return location with their initial way-in position, in order to deliver correct guidance notifications.

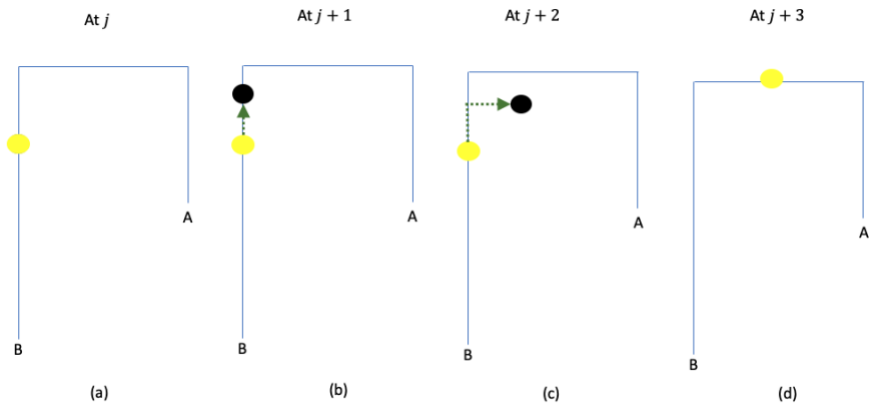
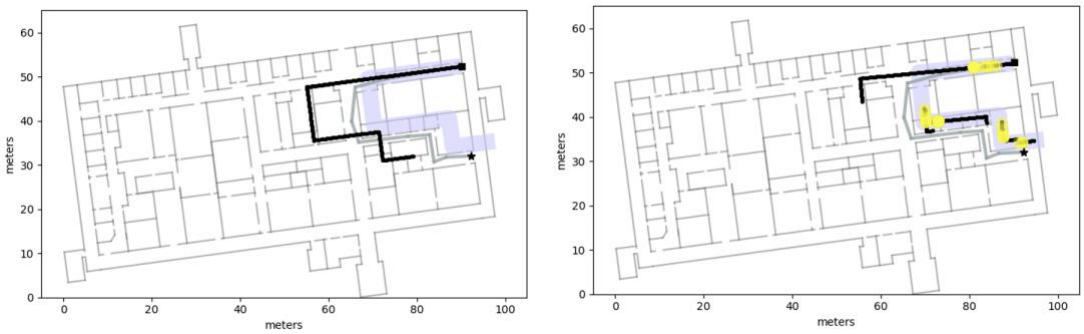


Figure 6.7: An example of projecting the user's position based on the last reliable position over a series of times during the return path. The blue line represents the way-in path from point A to B, while the return path begins at point B. The dotted green line is the projected path. The black dot is the projected position, and the yellow dot is the last reliable position. (a)In the beginning, the mapped point is also the last reliable position, so only one yellow dot is plotted. (b) The walker's position was projected based on the last reliable position. (c) The user turned right, and the return's projected position deviated from the way-in path. (d) A small mapping error(err_{map}) was observed and the mapped position is considered as the last reliable point. Eventually, the user reconverged back onto the original route.



(a) (b)

Figure 6.8: Examples of return path-matching using projected sequence(a) and hybrid matching(b). The way-in path is shown with a thick purple line, ending at the black square. The length of each segment is given by the number of steps recorded, multiplied by the step length measured during calibration. A gray line shows the actual path of the participant during the return phase. Projected sequences are shown with black lines. In

(b), reliable matches are shown as yellow circles. Note that in (a), the length of the initial segment appears to be longer than during the way-in, possibly because the walker took shorter steps, or took additional steps while looking for a place where to turn. In (b), the trajectory is corrected as soon as a new reliable match is found.

6.4 Dataset Description

Three categories of datasets are used to train the network for detecting the last reliable point. They are listed below, and Table 6.1 provides information on the dataset size in each dataset.

1. Actual way-in and return paths:

The paths were taken on the 3rd floor of the Engineering 2 building During the return journey, the walker intentionally made wrong turns. Consequently, samples corresponding to these off-route segments in the mapped index plot are also labeled as unreliable mappings. Figure 6.9 illustrates an example of such a path.

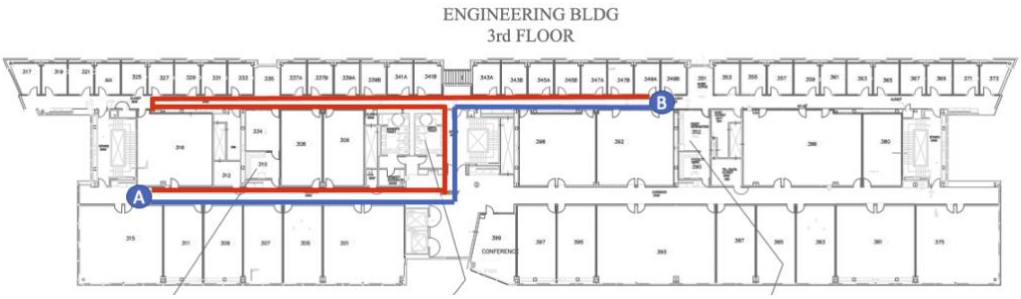


Figure 6.9: An example of dataset category 1. The trajectory of the walker. The blue line represents the way-in path from point A to point B, while the red line represents the return path from point B to point A.

2. Generative dataset created by concatenating segments in the buildings:

This is done by simulating actual way-in and return paths within Engineering 2 and Natural Science - Interdisciplinary Science buildings. Initially, we recorded magnetic field and steps/turns data in the segments, as illustrated in Figure 6.10. By concatenating these segments, we generated scenarios for both the way-in and return paths. In all paired paths (way-in and return), intentional off-route paths were added to create scenarios where the mapped index is incorrect. Figure 6.11 is an example of such a pair of paths in the E2 building; a way-in path is formed by concatenating segments S1-S2-S6, and the corresponding return path is S6-S2-S5-S5(reversed)-S1.

Although the generated pair paths may not exactly replicate the paths walked by individuals, this method still provides our model with sufficient data to learn the features necessary for identifying the reliability of the mapped index.

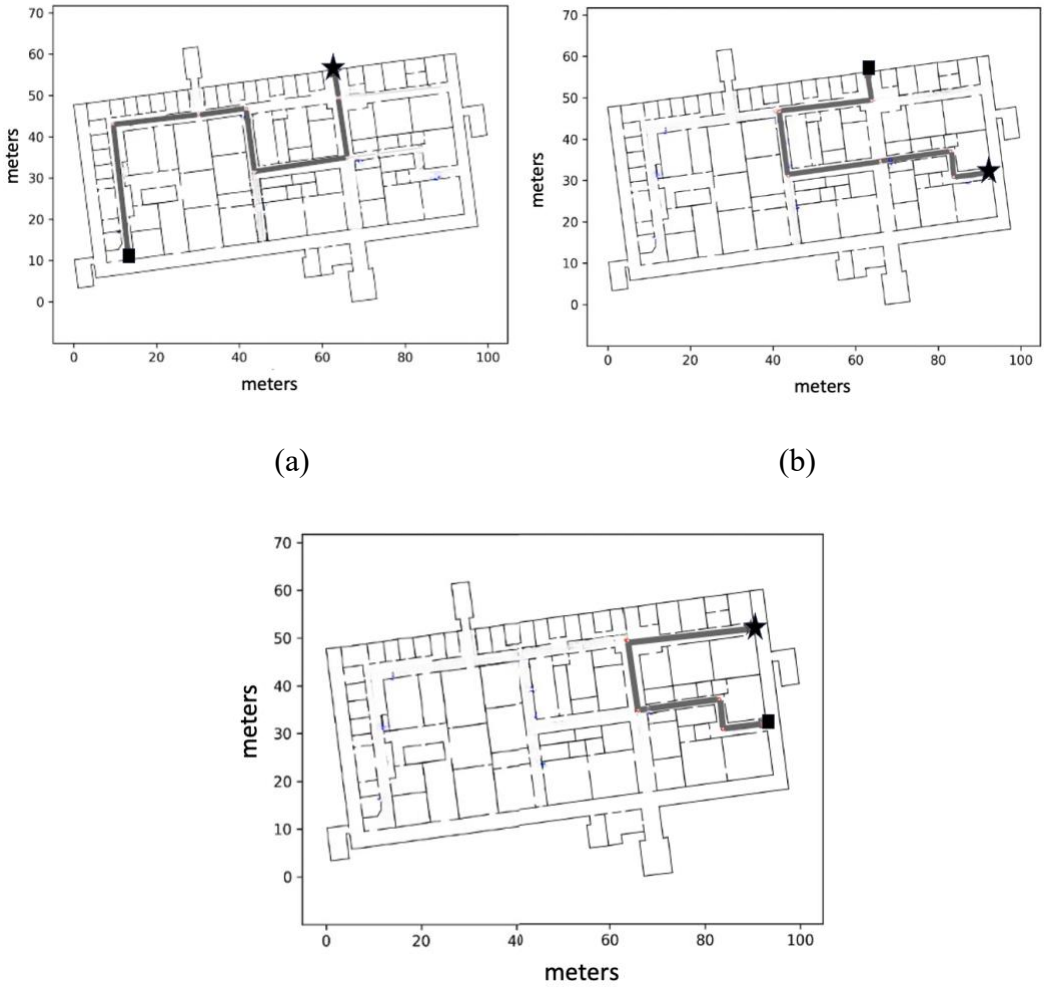
Figure 6.10: Segments in E2(top) and Natural Science - Interdisciplinary Science building (bottom). The red lines indicate the data (magnetic field and steps/turns) were recorded in these segments in the building.

Figure 6.11: An example of the generative dataset. The way-in route spans from point A to B, generated by connecting segments S1, S2, and S6. The return route, from point B back to A, includes an additional off-route segment, S5, appended. This return route is generated by connecting segments S6, S2, S5, S5 (reversed), and S1.

3. Seven visually impaired participants who used our app to navigate three paths in BE:

This is the dataset collected during the user study on the second floor of the BE building. Seven visually impaired participants were recruited for this

experiment by traveling three routes once (as the way-in phase). And then at the end of the third route, participants were instructed to retrace each route in reverse order (as the return phase). Each path contains 4 to 5 turns, with distances of 123m, 97m, and 72m, respectively. The defined routes are depicted in Figure 6.12, and the details of the data dataset are further discussed in the next chapter.



(a) (b)

(c)

Figure 6.12: Floor plan of the building with the tested paths highlighted. (a) R1 path; (b) R2 path; (c) R3 path. The tested paths are depicted in gray, with the start and end points indicated by a square and a star, respectively.

As previously discussed, the system estimates the user's location by aligning return samples with way-in samples, enabling the calculation of positioning errors based on misaligned samples. Consequently, the step length information is unnecessary. However, to calculate positioning error in meters, datasets in categories 1 and 2 utilize a fixed step length (0.54m, which is the average step length of the tester who collected the dataset) for user position calculation. For dataset category 3, tests were conducted concurrently with another WayFinding app [48] (developed by other PhD students in our lab, the details of the app will be discussed in Chapter 7) capable of estimating the user's step length in the initial step length calibration stage. Thus, the averaged step length measured from this app is applied to dataset category 3.

	Dataset # 1	Dataset # 2	Dataset # 3
# of paired paths	12	44	19
Size of data	4392	32428	9885
Number of turns	35	109	13

Table 6.1:Information of the dataset.

6.5 Error Metrics

Two error metrics are defined for the application: one is based on the generated trajectory, and the other is based on the accuracy of the generated LRP. This section provides detailed information about both metrics. It is important to highlight that we conducted a Leave-one-out cross-validation (LOOCV) methodology to evaluate the system's performance. The dataset from the user study (category 3) serves primarily for testing purposes because it is the actual dataset collected from seven visually impaired participants. During testing with LOOCV, the model was tested on the "**left-out**" participant while being trained using datasets from the **remaining participants in category 3 with the entire dataset in categories 1 and 2**.

6.5.1 Error Metric Based on the Ground Truth

The error is calculated based on the reconstructed trajectory by each method. Assuming the actual location of the user of the reconstructed trajectory based on the ground truth LRP_{gt} (defined in section 1.2.2) is represented as $(x_{gt,i}, y_{gt,i})$ and the calculated trajectory based on proposed methods, LRP_{local} and LRP_{NN} , are represented as $(\hat{x}_i, \hat{y}_i)$ where $\{i | i \in \mathbb{Z}, 1 \leq i < M\}$ }, and M is the number of steps in the path. The average error on each user's step can be calculated as follows.

$$E_{gt} = \frac{\sum_{i=1}^{M} ||(\hat{x}_i, \hat{y}_i) - (x_{gt,i}, y_{gt,i})||}{M} \quad (6.1)$$

6.5.2 Correctness of Predicting LRP

As mentioned in section 1.2.2, the ground truth LRP_{gt} was based on the optimally matched sequence $S(M)$ generated at the end of the return. LRP_{gt} serves as the reference for comparing LRP determined by other methods. The accuracy of LRP determined by different methods is defined using the following formula:

$$Accuracy = \frac{TP+TN}{total\ number\ of\ positions} \quad ,$$

Where:

$TP(true\ positive)$: The number of correctly identified LRP.

$TN(true\ nagative)$: The number of correctly identified non reliable positions.

1.3 Estimated position in different methods.

In this section, we compare different methods of finding the LRP and generating the user's estimated positions. The error based on the ground truth data, E_{gt}, is shown in Table 6.2. As mentioned earlier, the neural network models were trained by the whole dataset except for the data from the “left-out participant”. The left-out participant was only used for testing and was labeled in the first row of the table. It is important to note that in the last row of the table, the "default" method refers to considering all mapped points as reliable, which corresponds to the original path-matching algorithm described in Chapter 3. This default method is represented as $LRP_{Basedline} = \{all\ of\ mapped\ indices\}$. Additionally, for the "Linearly defined LRP" and "Baseline" methods, since no training is involved, the results for each participant column represent the corresponding test outcomes directly generated by the methods.

Left-out participant	P1	P2	P3	P4	P5	P6	P7
GAT	**0.56**	**1.6**	**0.90**	0.93	**0.74**	0.95	**0.35**
GCN	0.90	3.13	1.07	3.21	0.85	1.09	5.38
FCN	**0.53**	**1.05**	**0.68**	**0.51**	**0.62**	**0.24**	**0.22**
LSTM	1.15	3.12	0.62	1.12	1.50	1.21	2.65
CONV_1D	0.67	1.66	0.91	1.52	1.09	**0.7**	0.76
Linearly defined LRP	1.63	2.68	0.94	**0.85**	1.12	0.90	2.63
Baseline	0.82	3.20	1.52	3.64	1.98	1.6	3.62

Table 6.2: Error calculation (E_{gt} in meters) relative to ground truth. The two methods exhibiting the lowest errors for each left-out participant are highlighted in gray.

Test Path	P7, path#1	P2, path#3
GAT	0.63	**0.61**
FCN	**0.21**	0.64
Linearly defined LRP	4.98	7.63
Baseline	6.90	5.31

Table 6.3: Error calculation (E_{gt} in meters) relative to ground truth for the aborted cases in the user study.

As illustrated in Table 6.2, FCN demonstrates the highest performance, followed by GAT as the second-best model. Therefore, we have selected these two models for further comparison to assess their performance in challenging scenarios encountered during the user study, specifically the aborted cases that cannot be resolved simply by increasing the iDTW window size in the path-matching algorithm. The results of this comparison are presented in Table 6.3, indicating a significant reduction in errors achieved by both GAT and FCN in these difficult cases, allowing us to provide correct guidance to the user.

Regarding the correctness of predicting the LRP, Table 6.4 shows that the FCN network has the highest accuracy in most cases. When processing complicated paths, as shown in Table 6.5, both FCN and GAT can generate more accurate labels. However, FCN remains the better option across various scenarios.

Left-out participant	P1	P2	P3	P4	P5	P6	P7
GAT	0.75	0.76	0.87	0.81	0.82	0.70	**0.94**
GCN	0.75	0.77	0.77	0.73	0.70	0.74	0.83
FCN	**0.78**	**0.84**	**0.89**	**0.82**	**0.86**	**0.79**	0.82
LSTM	0.72	0.72	0.84	0.82	0.67	0.78	0.74
CONV_1D	0.74	0.81	0.88	0.79	0.75	0.77	0.79
Linearly defined LRP	0.63	0.79	0.78	0.81	0.82	0.79	0.83
Baseline	0.72	0.39	0.62	0.52	0.62	0.48	0.44

Table 6.4: The accuracy of predicting the labels of the last reliable positions by different methods. Note: for the "Linearly defined LRP" and "Baseline" methods, since no training is involved, the results for each participant column represent the corresponding test outcomes directly generated by the methods.

Test path	P7, path#1	P2, path#3
Linearly defined LRP	0.86	0.85
GAT	**0.98**	0.83
FCN	0.95	**0.86**
Baseline	0.15	0.20

Table 6.5: The accuracy of predicting the labels of the last reliable positions in the aborted cases in the user study.

Figure 6.13 illustrates reconstructed trajectories using method #1 (linearly defined LRP) or method #2(LRP generated by FCN). Near the coordinate (50,30), the walker made additional turns, and the system struggled to establish a reliable mapping in subsequent steps due to fluctuated magnetic fields. When reconstructing the path using linearly

defined LRPs, several incorrect LRPs (false positive LRPs) were identified over time, indicated by the green dots near the coordinate (50,15). The resulting trajectory (the green path in the lower part of the plot) shows significant misalignment with the expected trajectory (the thicker blue line).

In contrast, when utilizing LRPs determined by a neural network (FCN), no false positive LRPs were detected in Figure 6.13 (no black dot near the coordinate (50,15)). The LRPs detected near the destination are true positive LRPs. Additionally, the misalignment between the projected trajectory (the black line in the plot) and the expected trajectory (the blue line in the plot) is reduced. It is noted that although a few false positive LRPs were detected at the beginning of the path (near (50,30)) in both methods, they did not significantly affect the result.

Therefore, using the last reliable position provided by neural networks (particularly by FCN) reduces the error on constructed trajectory and increases the accuracy of identifying LRP. This approach proves advantageous for navigating complex paths, particularly in scenarios characterized by fluctuated magnetic fields.

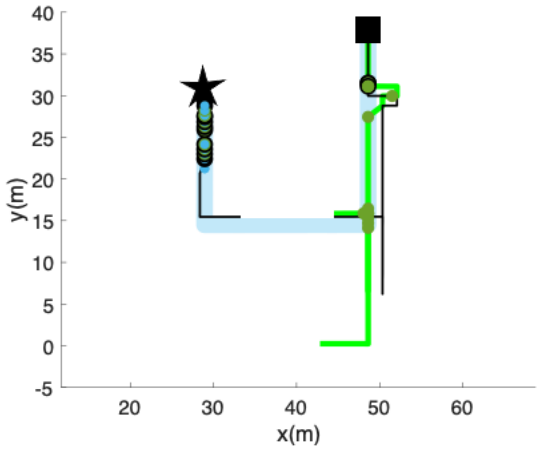


Figure 6.13: Projected trajectory based on the last reliable position determined by different methods. (Participant 7, Path#1 aborted case). The return path starts from a square and ends in a star. Thick blue line: expected trajectory. Black line: trajectory by method#2 (FCN). Green line: trajectory by method # 1(linearly defined LRP). Black and green solid circles: LRPs calculated by method#2 and method #1, respectively. Blue solid circles: ground truth for LRP.

6.6 Conclusion

In the original path-matching algorithm, the issue of changing the best matching sequences as more data becomes available during real-time navigation can lead to inconsistent mapping and confusing guidance, especially for users without visual cues. To address this, an improved algorithm called hybrid matching was discussed in this chapter.

The hybrid matching algorithm introduces the concept of finding a "last reliable position (LRP)" to determine the reliability of current mapped positions. Two methodologies for identifying LRP were proposed: one based on linear definitions and the other utilizing neural networks including fully-connected network (FCN), long

short-term memory (LSTM), 1D convolutional network, and graph neural networks (GCN and GAT).

A dataset consisting of three distinct collections from different buildings at UCSC was created to evaluate these methodologies. Three error metrics were defined to compare these methods and test the system's performance. The results revealed that using the FCN network to determine LRP yielded the best outcomes. It demonstrates the effectiveness of using neural network-based methodologies, particularly the FCN approach, to improve path-matching accuracy.

Chapter 7

Experiments with Backtracking Assistance

Navigating unfamiliar environments without a map presents significant challenges, particularly for individuals who need visual cues. Our experiments address this issue by implementing assisted return, a specific form of indoor navigation, to facilitate wayfinding for individuals with blindness. Introduced by Flores and Manduchi [21] assisted return involves providing support to guide blind users back to their starting point after traversing a specific path. Building upon this concept, we developed the SafeReturn app, utilizing path-matching algorithms to enable users to record and navigate routes with automatic guidance using their smartphones.

This chapter begins by introducing the user interface of the SafeReturn app, followed by a detailed explanation of the system's notification and navigation features for users. Furthermore, a user study involving seven visually impaired participants was conducted to evaluate the app's effectiveness, with the findings discussed in this chapter. It is important to note that the system implemented in the user study was based on section 6.3.1(Linearly defined LRP), where the user's last reliable position was linearly identified and utilized for navigation purposes.

7.1 User Interface Design

The SafeReturn app seamlessly integrates with VoiceOver [60], a gesture-based screen reader, to provide visually impaired users with an accessible interface on their iOS devices. Through VoiceOver, users can navigate the app's screens (storyboard) using sound cues and adjust settings to their preferences. Additionally, the app offers Apple Watch-based control, enabling users to interact with it and receive guidance while keeping their phone in their pocket.

Furthermore, the interface of SafeReturn is synchronized with another navigation app called "WayFinding[48]," developed by other PhD students in our lab . WayFinding is also designed for visually impaired individuals to navigate indoor environments where maps are available. By synchronizing the interfaces of these two apps, users can seamlessly switch between them based on the availability of building maps and their specific navigation requirements. This synchronization enhances the user experience by providing continuity in navigation across different environments.

The app's entry screen is depicted in Figure 7.1. The blue "Run Safe Return" button lets users start guidance on the main screen. Additionally, users can customize various settings by following steps:

1. Selecting their preferred guidance unit among foot, meter, or number of steps.
2. Entering their personalized average step length (it can be obtained by the WayFinding app in the initial calibration stage.)
3. Choosing between a 45^o or 90^o turn detector for navigation assistance.

It is recommended that the magnetometer be calibrated before starting the navigation task for the first time. This calibration process is initiated by tapping the yellow button at the top of the main screen, as detailed in the following section.

Figure 7.1 : The entry screen of SafeReturn.

7.2 Calibration of Magnetic Field

As noted in Chapter 4.2.1, the magnetometer is prone to significant drift due to hard iron or soft iron interference. Therefore, it is recommended that a one-time magnetometer calibration be performed before the user starts the navigation task. This calibration process can be quickly completed by rotating the smartphone along the x-y-z axes or using the conventional "∞" motion-based magnetometer bias calibration

method, as referenced in [17]. After calibration, the system will generate a fitted ellipsoid and display the residue of the fitted result, enabling users to determine if recalibration of the system is necessary. Figure 7.2 shows an example of the calibration screen.

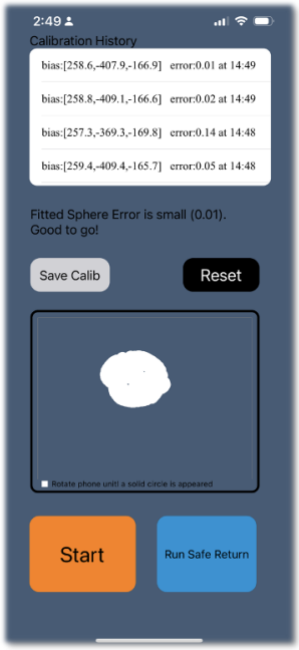


Figure 7.2 : The calibration screen of SafeReturn.

7.2.1 Main Screen of SafeReturn

The main screen, illustrated in Figure 7.3, includes a "Path Selection" scrollable list positioned at the top, presenting users with various path names for selection. Our application allows users to choose any path before starting their way-in journey without requiring prior knowledge of the selected path. However, during the return phase, users must select the same path name as their initial choice to ensure accurate matching with the corresponding way-in path within the system.

Below the "Path Selection" is the "Route Type" button, where users can specify whether the route is for way-in or return before starting the route. Another important button is located at the lower left corner to start or end the route.

Under the "Route Type" button, two figures are provided for real-time path-matching graph and trajectories visualization. It's important to note that these figures are solely intended for debugging by developers and are not accessible to the user.

As depicted in Figure 7.3, our design simplification emphasizes four primary interface components directly accessible to users via VoiceOver: "Path Selected," "Route Type," "Start/End," and "Repeat Notifications." The remaining setups, intended for debugging purposes, are configured once, and user intervention is not expected. This approach maintains simplicity in our interface, reducing disruptions for users. An example of the screen during the return phase is shown in Figure 7.4.

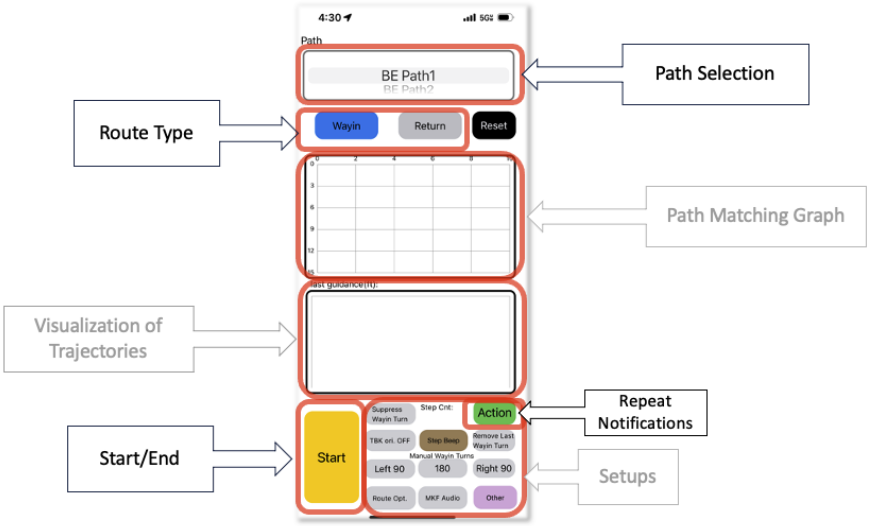


Figure 7.3 : The main screen of SafeReturn. Black notations: Primary interface components directly accessible to users. Gray notations: Components for debugging purposes or configured once in the system.

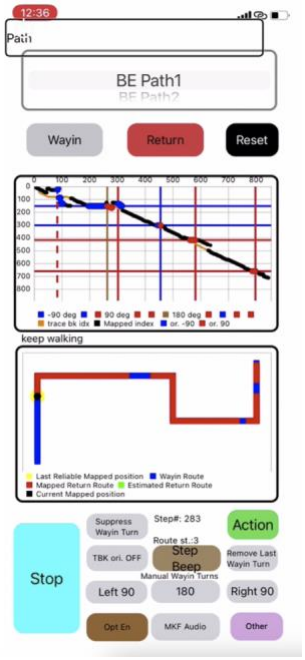


Figure 7.4 : Example of the main screen during the return phase. The path-matching graph (upper graph) displays colored lines indicating times when turns were detected during way-in (horizontal lines) and return (vertical lines). The visualization of

trajectories (lower graph) shows the real-time mapped position marked by a solid black circle, and the last reliable position is highlighted in yellow.

7.2.2 Watch Gestures for the System

As mentioned earlier, our goal is to create intuitive gestures for interacting with the app. To achieve this, we have developed an app for the Apple Watch, enabling users to keep their smartphones in their pockets while engaging with the system, as shown in Figure 7.5. Below are the primary gestures available in the Apple Watch app:

- Selecting the path: Users can navigate through the path options by swiping left or right on the Watch's face. VoiceOver will audibly announce the name of the currently selected item from the list .
- Starting the route: Initiating the route is as simple as rotating the Watch's crown in either direction until a distinctive "ding" sound is heard. This signals the starting of the route, accompanied by a notification: “Please start walking.”
- Ending the route: Similar to starting the route, users can end their journey by rotating the Watch's crown in either direction until they hear the familiar "ding" sound, indicating the conclusion of the route.
- Repeating last notification: Users can replay the previous notification issued by the app at any point during the path by performing a right swipe on the Watch's face.
- Hearing route description: To obtain a comprehensive description of the remaining route segments and turns from their current location to the

destination, users can perform a left swipe on the Watch's face at any time during their journey.

These gestures are designed to enhance user experience and facilitate easy navigation and control of the app while on the move.

Figure 7.5: Supported gestures in the Watch app. Before starting the path, users can swipe left or right on the Watch face to select a path. During navigation, swiping right will repeat the last notification, while swiping left will provide comprehensive route information.

7.2.3 Navigation Notifications - Turn-By-Turn Instructions

In our application, where map information is not available, the turn-by-turn instructions serve as the primary method of navigation assistance for visually impaired people. These instructions provide step-by-step guidance, helping users navigate themselves. These notifications are generated based on the current route and the user's location. Specifically, we calculate the distance of the user's location projected onto the associated route segment to the next turn point in the reversed way-in direction.

Figure 7.6 illustrates the different navigation states a walker may encounter. These states are described as follows:

- S0 state: This is the initial state of the user and serves as an intermediate state between notifications generated by other states. No notification is generated in this state.
- S1 state: This state indicates that the user has entered a new route segment. The notification generated in this state was originally "Walk straight for about XX [meters/feet/steps]. Then, turn left/right/approaching destination." However, during the user study from P4 to P7, the second sentence of the notification was removed. This adjustment aimed to prevent users from making another turn before reaching the actual turn point, enhancing navigation accuracy and user experience.
- S3 state: This indicates that the user is close to the next turn point or the destination (when the distance is smaller than Z steps). The notification generated in this state is "At the coming junction, turn left/right" or "Approaching destination."
- SW state: This state indicates that the user is walking in the wrong direction at a distance of more than Z steps from the way-in path. The path recovery notification is generated in this state: "You are walking in the wrong direction. Please turn around and start walking again."

The distance of Z steps (where Z =16 in our system) for defining the S3 and SW states was determined through trial and error in our initial experiments. With 16 steps defining the S3 state, the user is positioned approximately 8 meters away from the next turn junction, based on an average step length of 0.504 meters acquired during the user study. This distance provides sufficient time and distance for the user to prepare for or locate the next turn junction. Similarly, the number in the SW states indicates that the user has deviated from the correct path for a certain distance (8 meters), allowing the system to confirm whether the user is truly deviating from the correct path or if it's a temporary diversion. In cases of conflicting notifications, such as when the user enters a segment and an S1 notification is generated (while the audio is not finished), but they are immediately close to the next junction, an S3 notification will be generated. The ongoing notification is never interrupted. Additionally, it's important to note that the same notification is never repeated. This approach ensures a smooth and uninterrupted navigation experience for users, minimizing confusion and enhancing usability.

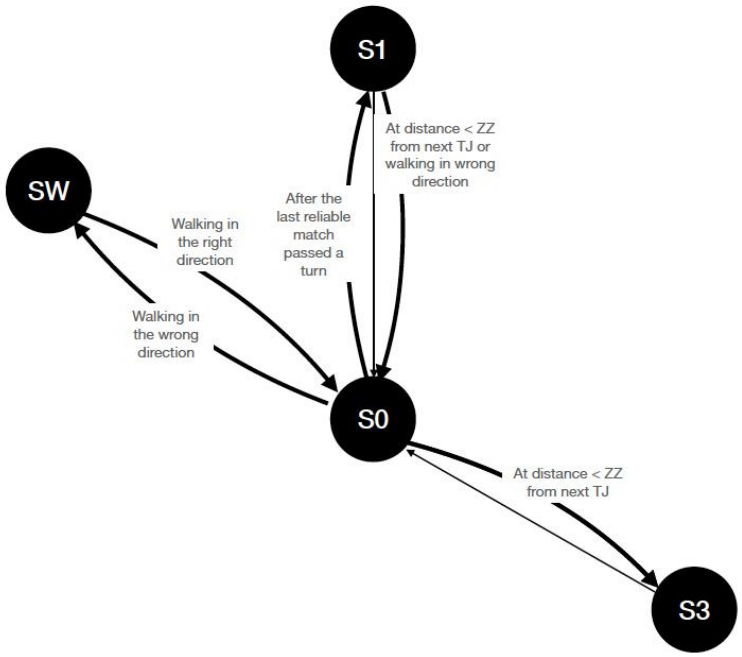


Figure 7.6: The state diagram for providing notifications.

7.3 Experiment – User Study

The user study was conducted concurrently for two apps: SafeReturn and WayFinding. The WayFinding app was developed by other PhD students in our lab (F. Elyasi and P. Ren) and they are also the developers of the step detector and turn detector in the SafeReturn app. The details about combining two apps in one user study are described in the next section. Seven participants were recruited for this experiment. Their characteristics are summarized in Table 7.1. All participants were blind with minimal to no light perception and were independent walkers. P6 recently transitioned from using a dog guide to a long cane and was still adjusting to it. P5 used hearing aids due

to hearing impairment. Everyone used iPhones except P7, who preferred a cell phone with a physical keypad. Only P1 wore a smartwatch (Apple Watch) regularly.

	Gender	Age	Blindness	Mobility aid	Preferred Unit	Phone preference	Smartwatch
P1	F	73	L	Dog	Steps	iPhone	Apple Watch
P2	M	69	B	Cane	Feet	iPhone	No
P3	M	53	B	Cane	Feet	iPhone	No
P4	F	69	B	Cane	Feet	iPhone	No
P5	M	75	L	Cane	Meters	iPhone	No
P6	F	76	L	Cane	Steps	iPhone	No
P7	F	72	L	Dog	Feet	Phone with keypad	No

Table 7.1: Characteristics of the participants in our study. For blindness onset, “B” indicates “since birth,” while `L' indicates “later in life.”

7.3.1 Experiment Setting

The experiment was conducted on the second floor of the Baskin Engineering building. Three routes (R1, R2, and R3) were selected, each comprising 4 to 5 turns, with distances of 123m, 97m, and 72m, respectively. The defined routes are depicted in Figure 7.7.

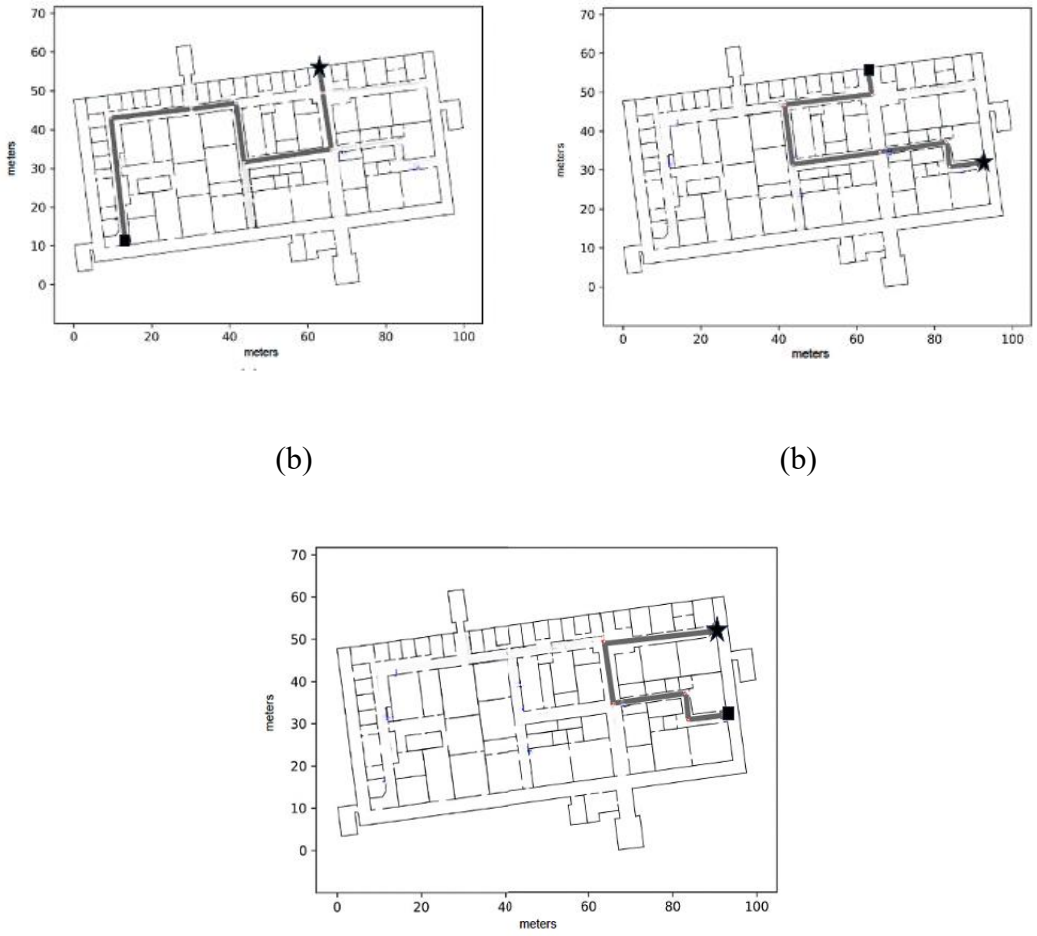


Figure 7.7 : The floor plan of the building with the tested paths is highlighted. (a) R1 path; (b) R2 path; (c) R3 path. The tested paths are depicted in gray, with the start and end points indicated by a square and a star, respectively.

In SafeReturn, a way-in route could be potentially traversed with the assistance of a sighted companion. In this study, instead of relying on a sighted companion during the way-in, participants used the WayFinding app as their guide. The WayFinding app is

specifically designed to assist visually impaired individuals in navigating from a starting point to a destination, utilizing map information of the building. It's important to note that SafeReturn does not utilize map information during the return phase.

This approach may introduce another level of difficulty, as participants had to navigate the way-in path with the assistance of the WayFinding app instead of relying on a sighted companion. This could lead to challenges such as navigating to a dummy route due to unfamiliarity with the WayFinding app or making additional turns due to swinging movements while attempting to locate a single turn point. However, by employing this approach, we demonstrated the system's ability to provide guidance in real-life situations, reflecting the complexities and challenges visually impaired individuals may encounter during navigation.

The way to combine two apps is as follows. First, a participant would use the WayFinding app to traverse three routes(R1-R2-R3), where the beginning of each route coincided with the end of the previous route. During this way-in phase, the SafeReturn app (running on a different iPhone, carried by the participant in a different pocket than the iPhone running the WayFinding app) recorded measurements (magnetic field, steps, turns) from each route. At the end of the third route, participants were instructed to retrace each route in reverse order, starting from R3 in the opposite direction. During this phase, participants received notifications from the SafeReturn app.

A separate building was utilized for the practice trial, where participants were introduced to both apps. The practice trial route was simpler, consisting of only two

turns with a total length of 63 meters. These two buildings were located close to each other.

7.3.2 Modalities

Experimental Procedure

The experimental protocol followed the guidelines approved by the University's Institutional Review Board, ensuring ethical conduct throughout the study. Prior to participation, each participant provided informed consent and received a comprehensive explanation of the applications' objectives and functionalities. Participants were encouraged to seek clarification on any uncertainties.

Special emphasis was placed on the notification system, ensuring participants were aware of upcoming turn alerts with advance notice. It was explained that upon receiving a notification, they were required to identify the nearest available turn, which could be in their proximity or a few meters down the way.

Initial Setup and Calibration

Following the introductory phase, participants underwent a simple calibration process outlined in 7.2. Subsequently, they were accompanied to the starting point of the practice trial. Each participant carried two iPhones in their pants pocket: one iPhone 12 running the WayFinding app, and the other an iPhone XR (running the SafeReturn app) recording way-in data. Participants also wore a wireless bone conduction headset

(Shokz OpenRun) to receive app notifications and an Apple Watch Series 8 to interact with the app.

Before starting the practice trial, settings such as VoiceOver speed and sound volume were adjusted to suit participant preferences. Additionally, participants were given the option to choose distance units for directions (meters, feet, or steps), with settings adjusted accordingly (see Table 7.1). It is important to note that the SafeReturn app initially provided distances solely in steps due to implementation oversight for the first three participants (from P1 to P3).

Practice Trial and Familiarization

Participants were guided through familiarization exercises, including practicing left and right swipes on the Watch interface. While all participants eventually mastered this, P2 initially struggled due to misinterpreting directional cues.

During the practice trial, the participants traveled a predefined route using the WayFinding app, followed by backtracking the path using the SafeReturn app. At the end of the practice trial, participants were asked about their preference regarding the sound of detected footsteps. All participants opted to retain the footstep sound, with some mentioning that it reassured them about the system's proper functioning.

(a) (b)

Figure 7.8: Participants interact with the Watch to start the test route.

Main Experiment Procedure

Upon completing practice trials, participants and experimenters relocated to the designated experiment site, starting from the initial route (R1) point. The sequential trials, detailed in section 7.3.1, were initiated. At the beginning of each trial, participants were positioned at the route's starting location and oriented in the initial

walking direction. They then selected the next route via the Watch interface and activated the app by rotating the crown, as shown in Figure 7.8.

Participants were instructed to swipe left on the Watch to hear a route description before navigation. Upon reaching the destination, participants stopped the app by rotating the Watch's crown. Optional rest periods were provided before starting subsequent trials, during which participants were repositioned at the next route's starting point (same as the previous route's endpoint) and correctly oriented.

Throughout the trials, experimenters maintained a safe distance from participants to avoid influencing routing decisions.

Post-Experiment Procedures

Following the final trial, participants and experimenters returned to the initial building, where participants completed a questionnaire comprising the ten System Usability Scale (SUS) questions and several open-ended inquiries [61].

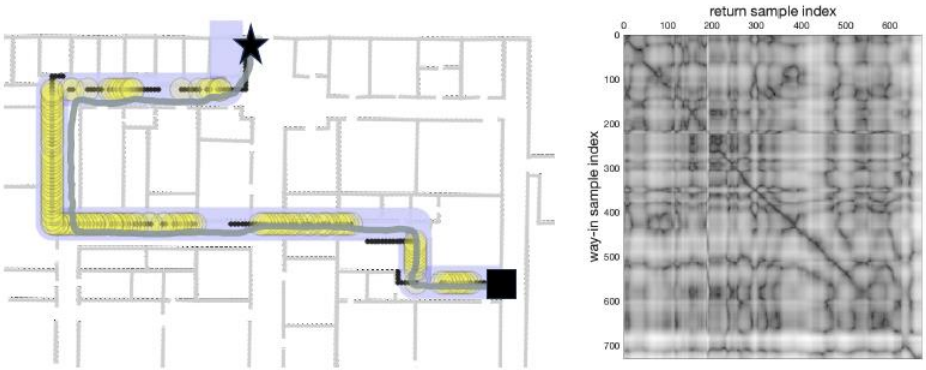
7.3.3 Observation and result

Successful Trials and Performance

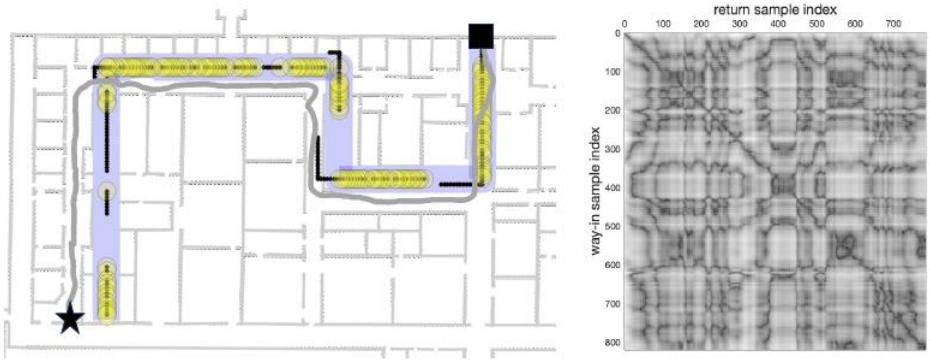
Figure 7.9 and Figure 7.10 show examples of successful trials with the SafeReturn app and Table 7.2 outlines the duration of these successful route traversals. In addition, Figure 7.11 demonstrates a situation where the participant took the wrong path. Then, the app was able to provide notifications for path recovery, and the participant walked back as directed by the app.

In these visualizations (Figure 7.9 and Figure 7.10), the way-in route (depicted with a thick purple line) consists of segments whose length is determined by multiplying the number of steps taken in each segment by the step length calculated during initial calibration. Consequently, these segments may not precisely align with the corridors depicted in the floor plan. However, this disparity does not affect the app's functionality, as the SafeReturn app aims to match the walker's location during the return with their location during the way-in, ensuring correct guidance notifications. Therefore, the metric consistency with the floor plan is not crucial for our purposes.

During the study, there were several situations in which the walker took extra loops during the way-in phase. Our way-in simplification method, discussed in section 3.1.1, effectively removed these loops. Figure 7.12 illustrates the result of the way-in simplification. While our method could remove the loops and preserve the path's geometry, there were instances where the length of the simplified way-in path differed significantly from the actual path length, as explained later in this section, resulting in the unsuccessful matching of the return path. Figure 7.12 (a) and (b) show the optimized way-in paths for both scenarios.

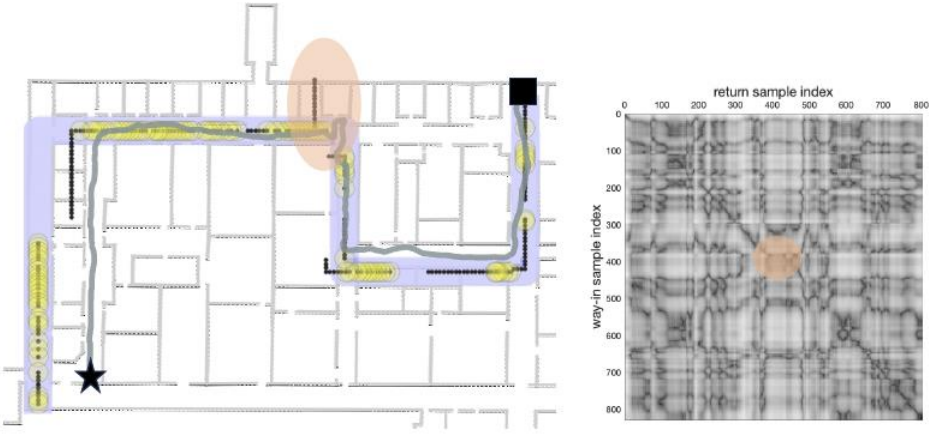


(a)

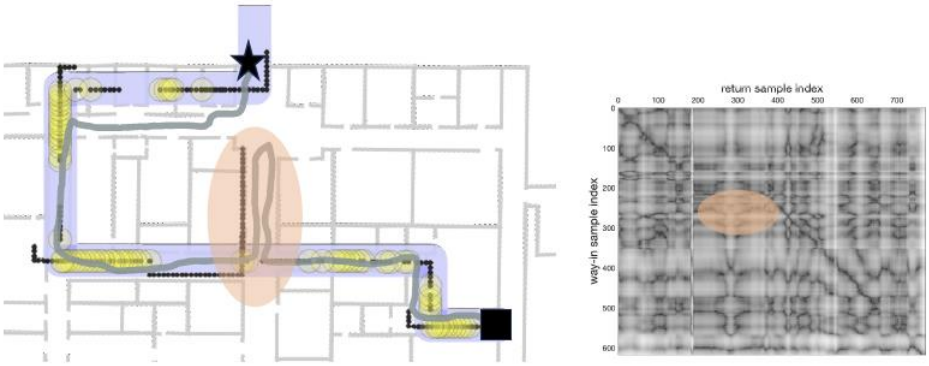


(b)

Figure 7.9: Examples of successful backtracking trials (hybrid matching). (a): Route R2 for participant P5. (b): R1 for P4. Left panel: The way-in path is shown with a thick purple line ending at the black square. The length of each segment is given by the number of steps recorded, multiplied by the step length measured during calibration. The actual path of the participant during the return phase is shown by a gray line. Reliable matches are shown as yellow circles. Projected sequences are shown with black lines. Right panel: Magnetic discrepancy for all pairs (i, j) of samples from way-in (vertical axis) and return (horizontal axis). Lighter gray indicates a larger discrepancy.

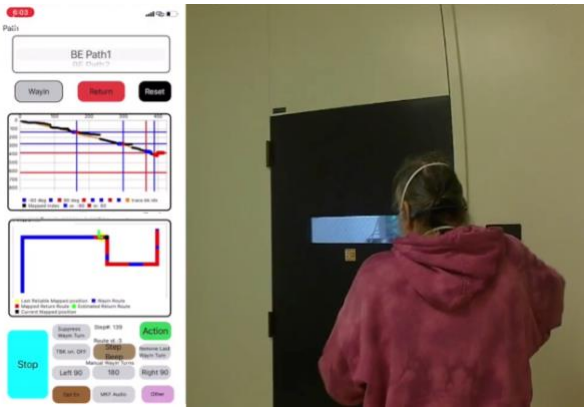


(a)

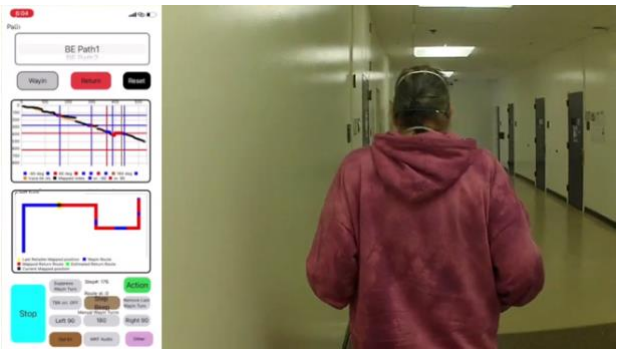


(b)

Figure 7.10: See caption of Figure 7.9 (a): path R1, participant P2. (b): path R2, participant P2. Highlighted are situations in which the participant took a wrong path and then walked back as directed by the app.

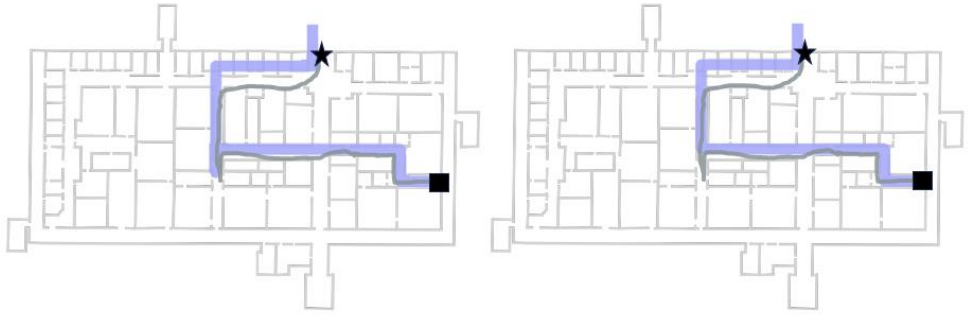


(a)

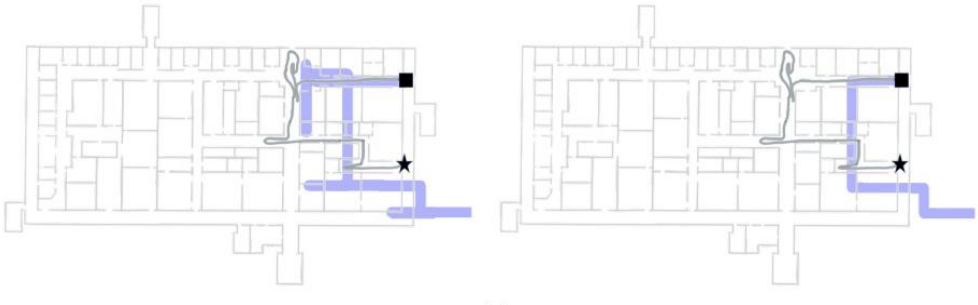


(b)

Figure 7.11 : Path recovery in the highlighted route for Figure 7.10 (a). Top: P2 was trapped in alcove path 1; Bottom: After receiving guidance from the system, P2 was able to walk back on the correct route.



(a)



(b)

Figure 7.12 : Examples of successful (a) and unsuccessful (b) way-in path simplification. The left panels depict the original way-in path with a thick purple line,

ending at the black square, alongside the approximate actual path taken by the walker (measured from the video), shown with a gray line. The right panels display the simplified way-in paths.

	P1	P2	P3	P4	P5	P6	P7	Length
R3B	180	x	115	134(E)	136	x	x	72m
R2B	182 (E)	232	x	238	173	154	149	97m
R1B	187	206	149	167	x,163(E)	184	x, x	123m

Table 7.2: Summary of the experiment for the WayFinding and SafeReturn routes. For successfully completed routes, we report the duration (in seconds). When displayed with a grey background, the participant missed one or more turns, or took a wrong turn, but was able to walk back and complete the route with guidance from the app. E: the route was completed, but verbal input from an experimenter was needed at some point. x: The trial had to be aborted due to the app's inability to track the participant. In two cases, a second attempt was made after a trial had been aborted.

On the other hand, six trials with the SafeReturn app were aborted due to tracking failures during the return route. Here is the analysis in each case:

- Participant P3 in path R2: The system alerted the participant about an upcoming turn ahead of time, but the participant made a wrong turn despite this. Although the app recognized this mistake and sent a "turnaround" notification to guide P3 back on track, he didn't follow it. P3 thought he knew the right way and continued walking in the wrong direction, believing he remembered the route. As he moved farther from the correct path, the app lost track of his location

because it mistakenly mapped P3's off-route position to an on-route position due to the similar magnetic field in different places.

- Participant P7 in path R1: P7, accompanied by a fast and confident dog, missed a turn due to her swift pace (Figure 7.14(a)). When she was notified to turn around, she was too far down the corridor. Although she attempted to turn around and return to the correct path, the system lost track of her due to the smaller size of the iDTW search window in the system's setup. Consequently, our current system has adopted a larger iDTW searching window setup to prevent similar occurrences in the future.

- Participant P5 in path R1: Initially unable to complete R1B, P5 veered off course and took multiple incorrect turns in an open space (Figure 7.14 (a)). After aborting the trial and starting a new one, P5 successfully completed the route by following the initial route description provided via the Watch interface.

- Participant P7 in path R3: P7 took several detours during the initial path, creating multiple loops. Figure 7.12(b) shows that our algorithm removed these loops and simplified the path while maintaining correct geometry. However, the first segment of the simplified path turned out to be significantly shorter than in the original path. This difference led to an unsuccessful backtracking trial.

- Participants P2 in path R3 and P6 in path R3: The main issue was a significant difference in the magnetic field readings. This issue is apparent in Figure 7.13, where the magnetic field discrepancies between pairs (i, j) of samples from the way-in (vertical axis) and return (horizontal axis) are displayed. A white horizontal line around the way-in sample index 120 and nearby samples indicates a significant difference in the magnetic field recorded at that location compared to any location during the return.

These analyses provide insights into the factors influencing trial outcomes and offer suggestions for enhancing the system's performance, as discussed at the end of this chapter.

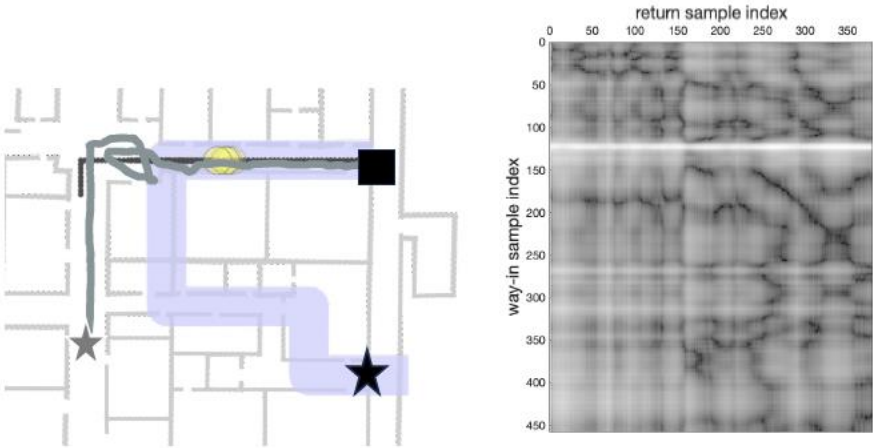
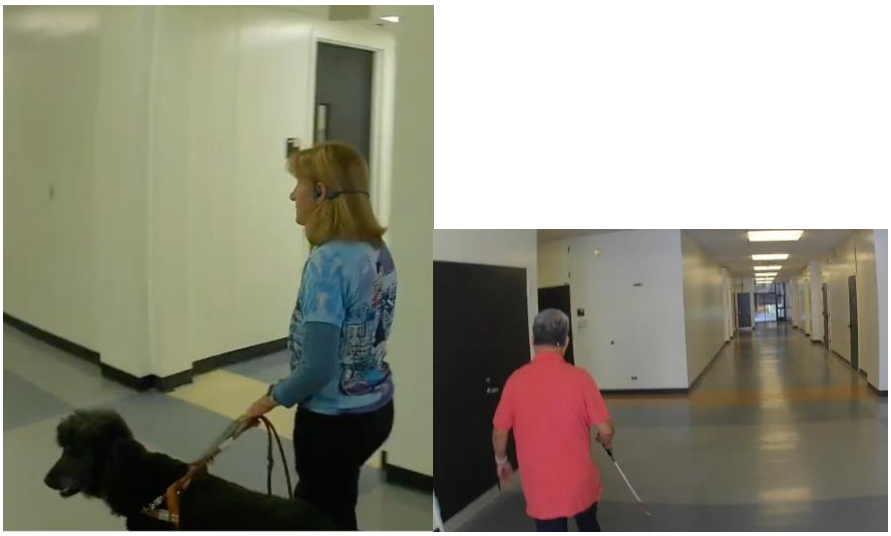


Figure 7.13: See caption of Figure 7.9. Path R3, participant P6. In this case, the app failed to track the participant. The gray star represents the point at which the trial was aborted; the black star is the desired destination.



(a) (b)

Figure 7.14: Pictures of our participants during the trial. (a): P7 ’s guided dog kept walking straight and missed the turn (path R1) (b): P5 veered off course on the first attempt in an open space (path R1)

7.3.4 Final Questionnaire and Open-Ended Questions

Table 7.3 shows the participants' responses to the System Usability Scale (SUS) questionnaire [48]. The overall score was 80.36, corresponding to a percentile rank of 90% based on the distribution of scores reported in [62]. It's noted that participants responded to both the SUS questions and open-ended questions for the WayFinding and SafeReturn apps rather than providing separate responses for each app.

	P1	P2	P3	P4	P5	P6	P7	Mean
1. I think that I would like to use this system frequently.	3	4	1	5	5	5	4	3.86
2. I found the system unnecessarily complex.	2	1	1	2	2	4	1	1.86
3. I thought the system was easy to use.	4	4	5	5	5	5	5	4.71

4. I think that I would need the support of a technical person to be able to use this system.	1	1	1	1	4	3	1	1.71
5. I found the various functions in this system were well integrated.	4	2	1	5	4	5	5	3.71
6. I thought there was too much inconsistency in this system.	3	3	1	1	2	4	1	2.14
7. I would imagine that most people would learn to use this system very quickly.	3	3	5	4	5	5	5	4.29
8. I found the system very cumbersome to use.	2	1	1	1	2	1	1	1.29
9. I felt very confident using the system.	4	4	3	5	4	5	5	4.29
10. I needed to learn a lot of things before	2	1	1	2	1	4	1	1.71

Table 7.3 : System Usability Scale (SUS) responses.

Below are the open-ended questions and a summary of the responses:

Do you think that the system always knew your location?

Most participants responded affirmatively, stating that the system accurately identified their location most of the time. However, both P2 and P3 answered "No." According to P2, there were instances when localization was inaccurate. Meanwhile, P3 noted that localization seemed to be influenced by individual walking styles.

Do you think that the system gave you the correct directions?

The majority of participants thought the system gave the right direction. However, P2 thought it was mostly accurate, similar to P1, who said it was right about 80% of the time.

The system often gives turning directions (such as "At the coming junction, turn right") with some advance notice, which means that you need to find the turn using your cane/dog. Was this a problem for you?

Participants generally stated that this was not a problem with the advance turning notifications, but some suggested they could be more consistent and accurate. For example, P4 mentioned a problem she had once with making a turn too early. P6 said it would be better if the notifications always came at the same distance from the junction. However, the system could not support this feature due to localization accuracy. P7 pointed out that this was the only part of the system that didn't meet her expectations for accuracy.

Were the notifications understandable? Too many notifications? Too few?

All participants found that the notifications were "fine" or "just right." P2 explained that it gave him an approximate distance to the next turn and alerted him just before the turn.

Was it easy for you to use the Watch?

Most participants found using the Watch easy, though P2 mentioned it required some practice to become accustomed to the gestures.

What would you like to have in this app that is not already there?

P5 mentioned the desire to know their current facing direction, which would be helpful when feeling lost. P3 suggested the option to scroll through route descriptions step by step, noting that the current implementation only allows a left swipe to view the remaining route without pausing to review each step.

Did you notice any difference between the WayFinding system and the BackTracking system (SafeReturn)?

Participants found both WayFinding and SafeReturn apps to be consistent, though P2 and P3 noted differences in the unit, which was later corrected.

Do you think that using this app would make you feel safer or more confident when traveling alone in a new place?

Participants expressed confidence in using the apps for traveling in unfamiliar places, highlighting benefits such as enhanced safety and reduced mental effort. P3, however, mentioned considering the SafeReturn app's potential use in navigating busy environments like conferences with numerous tables, where it could aid in returning to a specific location, such as a table after visiting the restroom. They discussed various

scenarios where the apps would improve navigation and boost confidence in diverse settings.

Overall, participants found the apps helpful for navigation, with positive feedback on providing the right directions and the ability to create mental maps of routes.

7.4 Discussion & Conclusion

We conducted a user study involving seven blind participants who provided invaluable real-world feedback on the system's performance. Overall, our SafeReturn app functions effectively when users adhere closely to the way-in path. However, the algorithm encounters challenges when users deviate significantly from the original route, leading to mismatches. Another issue arises from the spatial variability of the magnetic field within large corridors or hallways, which can cause mismatches when the user walks on a different trajectory within the same space during the return phase. Therefore, improvements are necessary to enhance the system's robustness in practical scenarios.

One essential improvement is to increase the system's iDTW window size. This adjustment will enable our path-matching algorithm to identify the mapped points after the walker returns to the correct path following significant detours. Additionally, given that the system implemented in the user study was based on section 6.3.1(Linearly defined LRP), integrating a neural network to detect the LRP could enhance location reliability, as discussed in section 6.3.2.

Moreover, ensuring that the walker's facing direction aligns correctly in open spaces is crucial for accurate navigation, especially at the beginning of the return phase. If the facing direction is misaligned with the hallway, the walker may struggle to locate the next turn points. One potential solution, which is currently being explored by another colleague in our lab (for the WayFinding app), involves using visual data, such as automatic landmark recognition [63], for sporadic "fixes" using computer vision techniques. In our assisted return application, this concept can be implemented as follows: at the end of the way-in path, the walker uses their smartphone camera to capture surroundings. Then, at the start of the return path, the system aligns the user's orientation by matching current surroundings with the previously captured visual information. After that, the user may move the smartphone back to their pocket and be tracked by our system.

Furthermore, during the process of simplifying the way-in path to remove redundant loops, it would be beneficial to acquire detailed step-length information for each step instead of using a fixed step length for all steps. This approach would result in a more realistic, simplified way-in route. Implementing a step-length estimator developed by another colleague in our lab can facilitate this enhancement [12].

By addressing these areas of improvement, we can enhance the functionality and usability of the system, ultimately providing a more reliable and seamless navigation experience for blind users in real-world environments.

Chapter 8

Conclusion

Wayfinding in an unfamiliar environment could be challenging and potentially unsafe for visually impaired people because it is difficult to recognize landmarks at a distance or any other visual information.

This thesis focuses on addressing this issue through the implementation of assisted return, a specific form of indoor navigation aimed at facilitating wayfinding for visually impaired individuals [21][7]. When a map of an indoor environment is not available, an assisted return system is designed for a visually impaired walker who has traversed a certain way-in route (possibly with the aid of a sighted companion) to traverse the same path in reverse (return).

We proposed a graph-based algorithm that leverages magnetic field data and inertial data (turns/steps) information to backtrack a walker's position when the map is unavailable (Chapter 3). Additionally, the algorithm addresses situations where the walker deviates from the intended path. Also, a straightforward approach to simplify the way-in route is introduced, which is suitable for some real-life scenarios where the walker might take extra loops during the way-in phase. We also investigated using the differences in magnetic fields between mapped locations (the "cost of magnetic field")

in our algorithm by studying the likelihood of the observed magnetic field (Chapter 4). Then, we tested the algorithm with the WeAllWalk dataset to compare the odometry systems based on steps/turns information. The analysis revealed that the $k \cdot 90^o+$ steps odometry provides better results (Chapter 5).

Furthermore, we developed an iOS app and conducted on-site testing to assess its performance. However, certain limitations arise in scenarios where the magnetic signature is unreliable. Therefore, we propose a hybrid matching approach by introducing the concept of LRP to enhance the system's performance and achieve more robust results (Chapter 6).

To evaluate these methods in real-world scenarios, we developed the SafeReturn app (based on the linearly defined LRP), featuring an intuitive interface integrated with the Apple Watch. We conducted a user study involving seven visually impaired participants (Chapter 7). The positive usability scores from the System Usability Scale (SUS) responses indicate overall satisfaction with the design. However, the fact that 6 out of 21 trials had to be aborted highlights several challenges in the current system. Consequently, we have proposed several improvements for the system

One challenge is to improve the system's ability to accurately track the user's position with similar magnetic fields across different locations. Implementing solutions such as using neural networks to detect a reliable mapping (i.e., detecting LRP) instead of the linearly defined LRP could significantly enhance location reliability, as discussed in section 6.3.1.

Another challenge is the requirement for users to begin a route from a specific starting point and maintain a predefined direction. To address this, a potential solution currently under investigation by another PhD student in our lab (for the WayFinding app) involves implementing periodic "adjustments" using computer vision techniques. This concept involves capturing the surroundings with a smartphone camera at the end of the way-in path and then aligning the user's orientation at the start of the return path using this visual information. Afterward, the user can put the smartphone back in their pocket and continue to be tracked by our system.

Furthermore, the system assumes that indoor environments consist of networks of corridors intersecting at certain angles (e.g., 90^o or 45^o). Its performance degrades when the intersecting angles deviate from these angles. Moreover, using the system in open spaces presents challenges, as magnetic field data is recorded for specific paths during the way-in phase, making it difficult to match magnetic sequences during the return phase when the walker's position is not confined to a specific corridor. Potential solutions to these challenges include placing a stronger emphasis on steps/turns information in open spaces. However, this approach may encounter issues if the walker's step length varies significantly. Alternatively, leveraging computer vision techniques to periodically adjust the walker's position using the camera can improve system performance in this situation. However, this adjustment process should be designed to minimize the need for the walker to hold the phone constantly. This approach has the potential to enhance the system's performance and usability.

Another challenging scenario during the user study was that users took too many extra loops in the way-in path. During the process of simplifying the way-in path to remove redundant loops, the simplified path sometimes became much shorter than the actual path, resulting in inaccurate way-in paths for us to match the return path to. As a result, we had to abort some of the tests for this reason. To resolve this issue, it would be beneficial to acquire detailed step-length information for each step instead of using a fixed step length for all steps. When users take extra loops during the way-in phase, this approach can generate more realistic and simplified routes. Adding a step-length estimator into the system developed by another PhD student in our lab can facilitate this enhancement [12].

Despite these challenges, positive feedback and SUS scores indicate that participants felt safer and more confident while using the app. It validates the potential of our proposed technology for providing assisted return for visually impaired individuals.

BIBLIOGRAPHY

[1] J. M. Loomis, R. L. Klatzky, R. G. Golledge, J. G. Cicinelli, J. W. Pellegrino, and P. A. Fry, "Nonvisual navigation by blind and sighted: assessment of path integration ability.," *J Exp Psychol Gen*, vol. 122, no. 1, p. 73, 1993.

[2] S. A. Cheraghi, V. Namboodiri, and L. Walker, "GuideBeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented," in *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, IEEE, 2017, pp. 121–130.

[3] R. Sammouda and A. Alrjoub, "Mobile blind navigation system using RFID," in *2015 Global Summit on Computer & Information Technology (GSCIT)*, IEEE, 2015, pp. 1–4.

[4] J. Hurtuk, J. Červeňák, M. Štancel, M. Hulič, and P. Fecil'ak, "Indoor navigation using IndoorAtlas library," in *2019 IEEE 17th International Symposium on Intelligent Systems and Informatics (SISY)*, IEEE, 2019, pp. 139–142.

[5] A. Budrionis, D. Plikynas, P. Daniušis, and A. Indrulionis, "Smartphone-based computer vision travelling aids for blind and visually impaired individuals: A systematic review," *Assistive Technology*, pp. 1–17, 2020.

[6] G. Fusco and J. M. Coughlan, "Indoor localization for visually impaired travelers using computer vision on a smartphone," in *Proceedings of the 17th International Web for All Conference*, 2020, pp. 1–11.

[7] C. Yoon *et al.*, “Leveraging augmented reality to create apps for people with visual disabilities: A case study in indoor navigation,” in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 2019, pp. 210–221.

[8] R. Crabb, S. A. Cheraghi, and J. M. Coughlan, “A Lightweight Approach to Localization for Blind and Visually Impaired Travelers,” *Sensors*, vol. 23, no. 5, p. 2701, 2023.

[9] Amanda Morris, “Navigational Apps for the Blind Could Have a Broader Appeal,” *The New York Times*, Dec. 20, 2021.

[10] D. Sato, U. Oh, K. Naito, H. Takagi, K. Kitani, and C. Asakawa, “Navcog3: An evaluation of a smartphone-based blind indoor navigation assistant with semantic features in a large-scale environment,” in *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, 2017, pp. 270–279.

[11] I. Abu Doush, S. Alshatnawi, A.-K. Al-Tamimi, B. Alhasan, and S. Hamasha, “ISAB: integrated indoor navigation system for the blind,” *Interact Comput*, vol. 29, no. 2, pp. 181–202, 2017.

[12] P. Ren, F. Elyasi, and R. Manduchi, “Smartphone-based inertial odometry for blind walkers,” *Sensors*, vol. 21, no. 12, p. 4033, 2021.

[13] I. Apostolopoulos, N. Fallah, E. Folmer, and K. E. Bekris, “Integrated online localization and navigation for people with visual impairments using smart

phones," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 3, no. 4, pp. 1–28, 2014.

[14] J. Zegarra Flores and R. Farcy, "Indoor navigation system for the visually impaired using one inertial measurement unit (IMU) and barometer to guide in the subway stations and commercial centers," in *Computers Helping People with Special Needs: 14th International Conference, ICCHP 2014, Paris, France, July 9-11, 2014, Proceedings, Part I 14*, Springer, 2014, pp. 411–418.

[15] T. H. Riehle, S. M. Anderson, P. A. Lichter, J. P. Condon, S. I. Sheikh, and D. S. Hedin, "Indoor waypoint navigation via magnetic anomalies," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2011, pp. 5315–5318.

[16] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 3146–3152.

[17] G. Ouyang and K. Abed-Meraim, "A survey of magnetic-field-based indoor localization," *Electronics (Basel)*, vol. 11, no. 6, p. 864, 2022.

[18] T. H. Riehle *et al.*, "Indoor magnetic navigation for the blind," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, 2012, pp. 1972–1975.

[19] Y. Shu, Z. Li, B. Karlsson, Y. Lin, T. Moscibroda, and K. Shin, "Incrementally-deployable indoor navigation with automatic trace generation," in *IEEE*

INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 2395–2403.

[20] N. A. Giudice, W. E. Whalen, T. H. Riehle, S. M. Anderson, and S. A. Doore, "Evaluation of an accessible, real-time, and infrastructure-free indoor navigation system by users who are blind in the mall of america," *J Vis Impair Blind*, vol. 113, no. 2, pp. 140–155, 2019.

[21] G. Flores and R. Manduchi, "Easy return: an app for indoor backtracking assistance," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[22] G. H. Flores and R. Manduchi, "Weallwalk: An annotated dataset of inertial sensor time series from blind walkers," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 11, no. 1, pp. 1–28, 2018.

[23] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang, "Indoor fingerprint positioning based on Wi-Fi: An overview," *ISPRS Int J Geoinf*, vol. 6, no. 5, p. 135, 2017.

[24] Z. Zuo, L. Liu, L. Zhang, and Y. Fang, "Indoor positioning based on bluetooth low-energy beacons adopting graph optimization," *Sensors*, vol. 18, no. 11, p. 3736, 2018.

[25] S. Sprager and M. B. Juric, "Inertial sensor-based gait recognition: A review," *Sensors*, vol. 15, no. 9, pp. 22089–22127, 2015.

[26] C. Fischer, P. T. Sukumar, and M. Hazas, "Tutorial: Implementing a pedestrian tracker using inertial sensors," *IEEE Pervasive Comput*, vol. 12, no. 2, pp. 17–27, 2012.

[27] Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, “A robust dead-reckoning pedestrian tracking system with low cost sensors,” in *2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, IEEE, 2011, pp. 222–230.

[28] M. Edel and E. Köppe, “An advanced method for pedestrian dead reckoning using BLSTM-RNNs,” in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, 2015, pp. 1–6.

[29] C. H. Tsai, P. Ren, F. Elyasi, and R. Manduchi, “Finding Your Way Back: Comparing Path Odometry Algorithms for Assisted Return,” in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, IEEE, 2021, pp. 117–122.

[30] S. Herath, H. Yan, and Y. Furukawa, “Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 3146–3152.

[31] B. Li, T. Gallagher, A. G. Dempster, and C. Rizos, “How feasible is the use of magnetic field alone for indoor positioning?,” in *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, 2012, pp. 1–9.

[32] W. Storms, J. Shockley, and J. Raquet, “Magnetic field navigation in an indoor environment,” in *2010 Ubiquitous Positioning Indoor Navigation and Location Based Service*, IEEE, 2010, pp. 1–10.

[33] J. Kuang, X. Niu, P. Zhang, and X. Chen, “Indoor positioning based on pedestrian dead reckoning and magnetic field matching for smartphones,” *Sensors*, vol. 18, no. 12, p. 4142, 2018.

[34] H. Liu, H. Xue, L. Zhao, D. Chen, Z. Peng, and G. Zhang, “MagLoc-AR: Magnetic-based Localization for Visual-free Augmented Reality in Large-scale Indoor Environments,” *IEEE Trans Vis Comput Graph*, 2023.

[35] R. Putta, M. Misra, and D. Kapoor, “Smartphone based indoor tracking using magnetic and indoor maps,” in *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, IEEE, 2015, pp. 1–6.

[36] J. Haverinen and A. Kemppainen, “Global indoor self-localization based on the ambient magnetic field,” *Rob Auton Syst*, vol. 57, no. 10, pp. 1028–1035, 2009.

[37] X. Fan, J. Wu, C. Long, and Y. Zhu, “Accurate and low-cost mobile indoor localization with 2-D magnetic fingerprints,” in *Proceedings of the First ACM Workshop on Mobile Crowdsensing Systems and Applications*, 2017, pp. 13–18.

[38] B. Gozick, K. P. Subbu, R. Dantu, and T. Maeshiro, “Magnetic maps for indoor navigation,” *IEEE Trans Instrum Meas*, vol. 60, no. 12, pp. 3883–3891, 2011.

[39] Y. Shu and B. F. Karlsson, “Path Guide: A New Approach to Indoor Navigation.” Jul. 2017. [Online]. Available: https://www.microsoft.com/en-us/research/publication/path-guide-a-new-approach-to-indoor-navigation/

[40] Y. Shu, C. Bo, G. Shen, C. Zhao, L. Li, and F. Zhao, “Magicol: Indoor localization using pervasive magnetic field and opportunistic WiFi sensing,”

IEEE Journal on Selected Areas in Communications, vol. 33, no. 7, pp. 1443–1457, 2015.

[41] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 31–36.

[42] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 889–898, 2011.

[43] P. Robertson, M. Angermann, and B. Krach, "Simultaneous localization and mapping for pedestrians using only foot-mounted accelerometers," in *Proccedings of ACM International Conference on Ubiquitous Computing*, 2009.

[44] S. Kaiser and E. M. Diaz, "PocketSLAM based on the principle of the FootSLAM algorithm," in *2015 International Conference on Localization and GNSS (ICL-GNSS)*, IEEE, 2015, pp. 1–5.

[45] R. Chen and J. S. Liu, "Mixture kalman filters," *J R Stat Soc Series B Stat Methodol*, vol. 62, no. 3, pp. 493–508, 2000.

[46] F. Elyasi and R. Manduchi, "Step length is a more reliable measurement than walking speed for pedestrian dead-reckoning," in *2023 13th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, 2023, pp. 1–6.

[47] B. Code, “CALIFORNIA Building Code,” *California Mechanical Code, California*, 2002.

[48] C. H. Tsai, F. Elyasi, P. Ren, and R. Manduchi, “All the Way There and Back: Inertial-Based, Phone-in-Pocket Indoor Wayfinding and Backtracking Apps for Blind Travelers,” *arXiv preprint arXiv:2401.08021*, 2024.

[49] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, “Fusion of smartphone motion sensors for physical activity recognition,” *Sensors*, vol. 14, no. 6, pp. 10146–10176, 2014.

[50] Apple, “deviceMotion,” Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/coremotion

[51] K. P. Subbu, B. Gozick, and R. Dantu, “LocateMe: Magnetic-fields-based indoor localization using smartphones,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, pp. 1–27, 2013.

[52] C. S. Kallie, P. R. Schrater, and G. E. Legge, “Variability in stepping direction explains the veering behavior of blind walkers.,” *J Exp Psychol Hum Percept Perform*, vol. 33, no. 1, p. 183, 2007.

[53] C. D. Fryar, M. D. Carroll, Q. Gu, J. Afful, and C. L. Ogden, “Anthropometric reference data for children and adults: United States, 2015-2018,” 2021.

[54] I. Ashraf, M. Kang, S. Hur, and Y. Park, “MINLOC: Magnetic field patterns-based indoor localization using convolutional neural networks,” *IEEE Access*, vol. 8, pp. 66213–66227, 2020.

[55] M. A. U. Shaikh, A. Mahmood, S. S. H. Zaidi, M. Zain, and M. Ashraf, “Future Position Estimation In case of GPS Outages,” in *2022 Third International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*, IEEE, 2022, pp. 1–6.

[56] B. Bonthu and M. Subaji, “An effective algorithm to overcome the practical hindrance for Wi-Fi based indoor positioning system,” *Open Computer Science*, vol. 10, no. 1, pp. 117–123, 2020.

[57] P. T. Mahida, S. Shahrestani, and H. Cheung, “Indoor positioning framework for visually impaired people using Internet of Things,” in *2019 13th International Conference on Sensing Technology (ICST)*, IEEE, 2019, pp. 1–6.

[58] J. Zhou *et al.*, “Graph neural networks: A review of methods and applications,” *AI open*, vol. 1, pp. 57–81, 2020.

[59] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[60] Apple, “VoiceOver,” Apple Developer Documentation. [Online]. Available: https://www.apple.com/voiceover/info/guide/_1121.html

[61] J. Brooke, “SUS: a retrospective,” *J Usability Stud*, vol. 8, no. 2, pp. 29–40, 2013.

[62] J. Sauro, *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability LLC, 2011.

[63] L. Chen, Y. Zou, Y. Chang, J. Liu, B. Lin, and Z. Zhu, “Multi-level scene modeling and matching for smartphone-based indoor localization,” in *2019*

IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), IEEE, 2019, pp. 311–316.

APPENDIX: PARAMETERS IN PATH-MATCHING ALGORITHM

Parameter Name	Description	Default
C_{ts} (e in app setup)	Turn suppression cost.	100
C_{mo} (d in app setup)	Mis-orientation cost.	5
$C_{sts1-off}$ (e2 in app setup)	Cost for changing status from on-route nodes to off-route nodes.	30
$C_{sts2-off}$ (e4 in app setup)	Cost for changing status from off-route nodes to off-route nodes.	8
$C_{sts-rev}$	Cost for changing status from off-route nodes to reversed-route nodes.	60
α (alpha in app setup)	For calculating $C_{MF\_off(i,j)}$.	2
mag_thres (mag_bound in app setup)	Threshold of the cost from the magnetic field for the off-route nodes.	40
Weight (Magnetic Field)	Weight for magnetic field during cost matrix calculation for path matching.	1/7.75
Path-Matching window size	This is the window size for path matching algorithm.	600