UNIVERSITY OF CALIFORNIA
SANTA CRUZ

PEDESTRIAN DEAD RECKONING USING SMARTPHONE INERTIAL DATA FOR BLIND WAYFINDING

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

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Fatemeh Elyasi

December 2023

The Dissertation of Fatemeh Elyasi
is approved:

Dr. Roberto Manduchi, Chair

Dr. Chen Qian

Dr. Yang Liu

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Pedestrian Dead Reckoning Using Smartphone Inertial Data for Blind Wayfinding

by

Fatemeh Elyasi

Accurate, robust, and infrastructure-free pedestrian positioning and navigation systems have gained significant attention in recent years due to their diverse applications. GPS is ineffective indoors and fixed infrastructure-based indoor navigation systems, such as beacons or Wi-Fi networks, pose practical and cost challenges. To address this, there's a growing demand for self-reliant navigation systems that seamlessly function indoors and outdoors. These systems often utilize sensor fusion and machine learning for precise and adaptive navigation. They can be integrated into standard smartphones, providing a portable and comprehensive navigation tool.

A specific beneficiary group for such systems includes individuals with visual impairments who rely on tools such as long canes or guide dogs. Self-reliant navigation systems can be customized for their specific needs, enhancing mobility and independence indoors and outdoors. Existing research has primarily focused on sighted individuals, however, there is an increasing interest to understand the unique challenges faced by visually impaired individuals and optimizing systems for more inclusive and effective solutions.

This dissertation addresses this need by developing a Pedestrian Dead Reckoning (PDR) system for inertial navigation using smartphones to assist visually impaired

individuals in indoor settings. Such PDR system requires two important components: step detection and step length estimation.

For step detection within our system, an LSTM-based network was developed, trained, and tested on the WeAllWalk data set, which includes inertial data gathered from ten blind and five sighted walkers. The achieved results on this data set surpassed existing benchmarks, highlighting the crucial role of selecting from the walker community for training data plays in determining results. Furthermore, the PDR system, incorporating this step detector method, outperformed the state-of-the-art learning-based model, RoNIN, in path reconstruction on the WeAllWalk data set.

For step length estimation, a model consisting of an LSTM layer followed by four fully connected layers was implemented. The same network scheme was used to predict either step length or walking speed (allowing for integration over a step period to calculate step length). In the initial step, data was collected from twelve sighted participants who traversed four routes with varying stride lengths. Results from sighted participants suggest that step length can be predicted more reliably than average walking speed over each step. Subsequently, the model was trained and tested on data from seven blind participants. The obtained results highlighted the different gait patterns among sighted and blind walkers, emphasizing the importance of designing systems for assisted navigation based on data from the visually impaired community.

Finally, an iOS application named WayFinding was designed to aid indoor navigation for blind travelers. The developed step detector module was integrated into this app. However, for this study, a calibrated step length was used instead of the

step length estimator. WayFinding enables an individual to determine and follow a route through building corridors to reach a certain destination, assuming the app has access to the building's floor plan. This app exclusively utilizes the inertial sensors of the smartphone, requiring no infrastructure modifications, such as the installation and support of BLE beacons. A watch-based user interface and speech-based notifications enable hands-free interaction for blind users. A user study involving seven blind participants was conducted in our campus buildings to assess the system's performance. All participants successfully navigated the pre-defined routes and provided positive feedback during the post-experiment interviews and questionnaires.

Acknowledgments

I extend my deepest gratitude to my advisor, Dr. Roberto Manduchi, for his unwavering support, guidance, and motivation throughout my academic journey. His mentorship and supervision have been invaluable in shaping this study. I'm also grateful to Dr. Chen Qian and Dr. Yang Liu for agreeing to be on my committee, reviewing my thesis, and offering valuable advice.

My heartfelt appreciation goes to my team members, Peng Ren and Chia Hsuan Tsai, involved in this project. Our collaborative efforts and shared determination in crafting assistive technology have been truly inspiring. Additionally, I extend sincere thanks to all the participants who generously dedicated their time and provided invaluable feedback during the data collection and user study, greatly enriching this research.

A special note of appreciation goes to my husband, Hossein Hodaei, whose constant encouragement and support have been pivotal to my perseverance. His unwavering belief in me has been a driving force, for which I am endlessly grateful.

I'd also like to recognize my parents, sister, and brother for their patience and support throughout my academic pursuits, even from afar.

Chapter 1

Introduction

While the fields of localization and odometry have traditionally focused on moving platforms, there is a growing interest in developing mechanisms that can effectively localize and track human pedestrians as they move, especially within indoor environments. This emerging interest arises from a diverse range of applications, each presenting unique benefits and challenges.

Recently, localization and tracking technologies have played a crucial role in addressing challenges related to contagious diseases and interpersonal interactions. These technologies are now essential for accurately tracking interpersonal contagion in close-contact scenarios. Innovative methods have been developed for this purpose [63, 79, 75]. Another application involves monitoring people's movements in various contexts, including healthcare facilities. This is crucial for ensuring patient safety and well-being, potentially enhancing the quality of care [90, 108]. Another important application is providing navigation and location-based information to individuals exploring unfamiliar places,

particularly benefiting those with visual impairments. Visual challenges, such as the inability to recognize landmarks, preview route sections, and access visual maps, are common among individuals with visual impairments. While some blind individuals can construct mental maps through direct locomotion experience, others may have a limited grasp of their surroundings during route navigation [106, 80]. Technological solutions supporting safe navigation for the visually impaired individuals can significantly enhance their mobility and independence, offering valuable assistance [3].

Developing an indoor pedestrian localization and navigation system is a complex task. Some approaches necessitate the setup of dedicated infrastructure, like BLE beacons [26] or UWB beacons [36], which limits their practicality to locations where these devices have been specifically deployed. Alternatively, certain methods rely on external signals that are expected to be accessible at the area of interest, such as GPS measurements, radio signals from Wi-Fi routers [16], and magnetic field fluctuations caused by electrical appliances [97]. GPS-based localization is ineffective indoors, and methods relying on Wi-Fi signals or magnetic fields necessitate an initial calibration phase (commonly known as fingerprinting), making them unstable over time, especially when Wi-Fi routers are repositioned.

Two indoor pedestrian tracking methods that do not require external infrastructure or prior calibration are visual odometry and inertial odometry. Inertial sensors and cameras are commonly found in smartphones, but visual odometry is power-intensive, needs clear field of view, and raises privacy concerns. Despite known issues like drift, inertial odometry is argued to be more practical for pedestrian tracking than visual-based

methods. Users can conveniently place their smartphone in their pocket and rely on inertial sensors, making it suitable for visually impaired individuals who may use a cane or a guide dog while walking.

Standard strap-down inertial odometry, which involves double integrating accelerometer data while subtracting the gravity vector, faces challenges like drift caused by noise and bias. It uses gyro data to track the gravity direction but computes odometry in the sensor's frame, making it prone to misinterpretations of walking direction due to smartphone orientation changes relative to the user's body. To improve performance, we can leverage the specific characteristics of human walking. Walking consists of steps with stance and swing phases. Inertial sensors on the walker's feet reset estimated velocity during stance, reducing drift (known as zero-velocity updates or ZUPTs [33]). These are effective for foot-mounted sensors but not applicable to smartphones. Modern methods employ neural networks to learn the dynamics of inertial data recorded during human walking, resulting in reduced odometry drift. These machine learning approaches require well-calibrated datasets containing both inertial data and ground truth positions [14, 45, 66, 99].

Pedestrian Dead Reckoning (PDR) determines the user's position by detecting step events and adjusting the location based on step length and orientation. PDR is favored for its simplicity and robustness. Errors in step length determination have a linear impact on the computed location, whereas uncompensated accelerometer bias leads to a quadratic effect due to double integration. Machine learning methods are employed to estimate step lengths from inertial data [40, 110, 55, 116], offering improved reliability

compared to previous heuristic techniques [61, 111, 54].

Inertial navigation systems can greatly benefit blind travelers who face significant wayfinding challenges due to their visual impairment. These systems have the potential to enhance their safety and confidence during independent travel. Existing inertial odometry systems, mainly designed for sighted individuals, may not be suitable for blind walkers. This is because the gait patterns of blind individuals using a cane or guide dog differ from those of sighted walkers [44]. For example, blind walkers often exhibit varying heading directions due to side-to-side swings with a long cane. Navigating without sight can lead to frequent encounters with obstacles and other people, necessitating frequent stops and reorientation. These occurrences introduce inaccurate inertial measurements that may pose challenges for odometry systems designed for sighted individuals.

To create an odometry system for individuals with visual impairment, it's crucial to include data from blind walkers, especially when using learning-based methods. The WeAllWalk data set, previously compiled by Dr. Manduchi's research group, is the only publicly available data set with inertial data from blind participants. They carried two iPhones and traversed various paths in two different buildings, with blind walkers using canes or guide dogs as travel aids.

This dissertation focuses on developing a smartphone-based indoor inertial localization and odometry system designed to assist visually impaired individuals in navigating indoor environments. In situations where a building map is accessible, the system is equipped for wayfinding, delivering detailed turn-by-turn routing information and instructions. This application was implemented in collaboration with another Ph.D.

researcher in our group

To accomplish this goal, a Pedestrian Dead Reckoning (PDR) system was developed, relying on step-detection algorithms using phone inertial data. First, in Chapter 2, I provide a background on positioning techniques. Chapter 3 involves the design, training, and testing of a Long Short-Term Memory (LSTM)-based model. This model, leveraging the WeAllWalk data set for step counting, not only outperforms existing benchmarks but also sets a new standard for accuracy on this data set. Furthermore, the developed PDR system comprises multiple components, including the aforementioned step counter model, a turn-detector module (crafted by a fellow Ph.D. researcher), and a learned fixed step length. This comprehensive system has undergone rigorous testing on the WeAllWalk data set, yielding results that surpass those achieved by the leading learning-based odometry system, RoNIN. These results mark a significant milestone in the development of our indoor inertial localization and odometry system.

In Chapter 4, I present another essential component of a PDR system involving the estimation of step length. To enable learning about step lengths, a new data set was gathered from twelve sighted participants. This was important to address the accelerometer range limitation of the foot sensor in the WeAllWalk data set. For this data set, participants traversed four routes with different stride lengths, during which inertial data was collected using two smartphones worn on their bodies. Ground truth step lengths, along with average walking speed during each step, were established using foot-mounted sensors and EKF-based odometry integrated with zero-velocity updates. Subsequently, an LSTM network was introduced to predict either step length or average walking speed

during a step. The experimental results robustly indicated that predicting step length is more reliable for achieving greater accuracy in distance measurements compared to estimating the walker's velocity. Finally, the performance of the proposed step length estimator model was evaluated using data collected from seven blind participants. The results emphasized that a model trained exclusively with sighted walkers consistently yielded poor results when tested with blind users. However, when the training set included data from this specific community, the results exhibited substantial improvement.

Finally, in Chapter 5, WayFinding is introduced, an iOS-based application designed to assist blind travelers in navigating indoor environments. Assuming access to the building's floor plan, the system utilizes the phone's inertial data to provide detailed turn-by-turn navigational instructions, guiding blind travelers to their destination. The system integrates two PDR-based localization technologies with particle filtering for inertial-based localization. One approach incorporates the previously mentioned step detector module, identifying steps with a length equal to the estimated step length and direction derived from the phone's azimuth angle. Since the step length estimator model was trained on sighted walkers, we employ a calibrated adaptable step length. The second PDR technology is based on RoNIN [45], estimating position by integrating the estimated velocity over time. Moreover, a user-friendly speech-based interface has been incorporated for blind travelers, allowing interaction through a smartwatch. Users receive instructions and feedback via speech, and notifications are strategically designed to provide ample time for decision-making while addressing the inherent inaccuracies associated with the dead-reckoning nature of inertial-based localization. Lastly, a user study involving seven

blind travelers was conducted to assess the application's performance in real-life scenarios. Participants traversed four routes in two different buildings, including one to become familiar with the app usage. Observations and feedback obtained through interviews with the participants were encouraging, indicating that the system can be a feasible aid for navigating indoor spaces.

Chapter 2

Related Work

Navigation is the process of guiding an object or a person from its current location to the desired destination. This task involves two main components: positioning and guidance. Indoor navigation systems find applications in various scenarios. For example, providing the position of emergency responders like firefighters and police officers in rescue efforts aids navigation in challenging environments, such as smoke-filled buildings. Another crucial application is wayfinding for individuals in office buildings, shopping malls, and transit hubs, which is particularly challenging for blind travelers and requires support for independent travel.

Positioning techniques can be broadly categorized into two main types based on the technologies they utilize: external infrastructure-based systems and infrastructure-free systems [95].

2.1 Infrastructure-Dependent Systems

2.1.1 WiFi-based systems

WiFi, a wireless local area network (WLAN) technology, is extensively utilized indoors, making it well-suited for indoor positioning applications. Two primary methods are commonly used for positioning with WiFi signals: signal strength-based methods and fingerprinting.

2.1.1.1 Signal strength-based

In this approach, the method involves measuring the received signal strength index (RSSI) of each WiFi access point (AP) to compute the distance between a target device and multiple APs. Employing trilateration techniques, the system calculates the target device's position in relation to the known positions of these APs. However, it is worth noting that the reliability of this positioning technique is challenged by the presence of strong reflections and scattering conditions indoors, leading to a significant reduction in the accuracy of RSSI measurements [68, 18].

2.1.1.2 Fingerprinting

This method operates in two stages: offline mode and runtime mode. In offline mode, a database is generated, creating a radio map that depicts the relationship between RSSI values and positions at reference points with known locations. During runtime, the measured RSSI is compared with stored RSSI values in the database to

estimate the device's position. However, this technique also suffers from the signal attenuation issues. Additionally, constructing and updating the fingerprint database is a time-consuming process. It's important to note that mobile devices, being small and constrained by battery power, present challenges in minimizing power consumption in these methods [68, 18, 109].

2.1.2 Bluetooth-based systems

Using Bluetooth for positioning purposes has been extensively investigated in the past. The primary challenge in these systems lies in the considerable time required to obtain a sufficient number of nearby Bluetooth devices, leading to a significant increase in localization latency. In 2010, the situation changed with the introduction of Bluetooth 4.0, which includes Bluetooth Low Energy (BLE). Compared to classic Bluetooth, BLE offers improved data rate, coverage range, and higher energy efficiency. Similar to WiFi-based systems, Bluetooth-based systems rely on Received Signal Strength Indicator (RSSI). The methods used for positioning with BLE beacons can be divided into two main categories: distance-based and fingerprinting-based. In the distance-based approach, knowledge of the exact locations of BLE beacon stations is essential. Consequently, the target's position can be estimated using trilateration. The fingerprinting-based approach is similar to WiFi-based fingerprinting methods. The Reference Fingerprint Map (RFM) is constructed during offline mode, and in the online phase, the position can be estimated by comparing the collected fingerprint with the fingerprints in the RFM. Although it has been demonstrated that BLE is more accurate than WiFi in identical locations, it

shares similar drawbacks, as mentioned in the previous section. Creating an RFM is time and energy-consuming, requiring periodic updates due to possible changes in the environment[9, 18, 58, 122].

2.1.3 Ultrasound-based systems

Researchers drew inspiration from bats, which use ultrasound signals for night-time navigation, leading to the exploration of ultrasound signals in positioning systems. To implement such a system, ultrasonic receivers are strategically placed in known locations within the environment. The mobile target acts as a transmitter, emitting both radio and ultrasonic signals simultaneously. Given that radio signals travel much faster than ultrasound signals, receivers can synchronize quickly. By measuring the time difference of arrival between the radio and corresponding ultrasonic signals, each receiver can infer its distance from the transmitter. Trilateration, using multiple distances to known locations, then allows for the determination of the target's position. However, these systems face challenges due to potential inaccuracies caused by reflections, obstacles, and multi path receptions. Additionally, the installation of several receivers is costly and complex, and the speed of ultrasound signals can be affected by temperature changes, introducing further measurement challenges [72, 41, 68].

2.2 Infrastructure-Free Systems

2.2.1 Vision-based systems

Visual odometry, employing techniques like Structure from Motion or visual SLAM, reconstructs the camera's position, orientation, and world structure by continuously tracking features across frames [24]. Frameworks such as Apple ARKit integrate visual odometry with inertial odometry to precisely determine the camera's global position and orientation [13]. Scale determination often relies on data from acceleration or depth sensors. Leveraging visual data for indoor localization holds significant promise. While individuals with sight heavily rely on their eyes to gather essential spatial information, mobile targets can also be equipped with cameras, like those attached to robots or smartphones carried by people as sensors. These systems execute localization by tracking environmental features. However, vision-based positioning systems, reliant on cameras as sensors, have limitations. They may not perform optimally in low-light environments, when the camera is obstructed, or lacks a clear line of sight, for instance, when a user keeps a phone in their pocket [89, 68, 18, 34].

2.2.2 Inertial sensor-based systems

Inertial sensors, typically comprised of accelerometers and gyroscopes integrated into Inertial Measurement Units (IMUs),are significant components within the broader category of Micro Electro Mechanical Systems (MEMS).The changes in inertial measurement signals captured by an IMU during body movement are employed in the positioning

process. IMU sensors can be categorized into two types: wearable sensors and portable sensors. Wearable sensors can be attached to one or more parts of the user's body, while portable sensors can be integrated into devices like smartphones or smartwatches, which may be placed in the user's pocket or held in hand during motion. In contrast to other sensor types, inertial sensors offer several advantages, ncluding small size, lightweight design, portability, wearability, cost-effectiveness, low power consumption, and independance from infrastructure [96, 95]. The subsequent section explores the primary methods that leverage inertial sensors for various applications.

2.2.2.1 Strapdown Inertial Navigation Systems (SINS)

The Inertial Navigation System (INS) integrates the sensor outputs to calculate the navigation solution (velocity, position, and attitude) continuously. An unaided inertial navigation system suffers from integration drift, i.e. small errors in acceleration and angular rates cause errors in speed and heading direction, which accumulate over time and form a drastically larger error in the position estimate. Inertial sensors can be strapped to a moving object to create a strapdown inertial navigation system. In this scheme, the sensor outputs are measured in the sensor frame rather than the navigation frame. The attitude is calculated by integrating the angular rates measured by the three gyros. Next, this orientation is used to transform the acceleration values measured in the sensor frame into the navigation frame, and the gravitational component is removed from the z-acceleration. The acceleration measurements in the navigation frame are then integrated to obtain the velocity. Finally, the velocity is integrated to obtain the position

(see Figure 2.1).

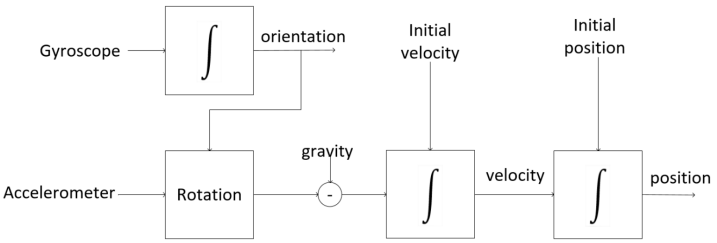


Figure 2.1: Strapdown Inertial Navigation System [115]

Foot-mounted sensors are very popular among strapdown inertial navigation systems for use in indoor positioning. Due to the presence of zero velocity conditions during stance phases, the positions can be regularly corrected using techniques such as Zero velocity UPdaTes (ZUPT). The pedestrian's foot alternates between a moving stride phase and a stationary stance phase when the foot touches the ground. ZUPT is applied during the stance phases where the velocity is theoretically zero in order to limit the growth of the velocity and, consequently, the position errors in inertial navigation systems. Moreover, Kalman Filter-based frameworks can be implemented to combine zero velocity measurements with inertial navigation system (INS) solutions. To further improve the position accuracy, it has been shown that instead of simply resetting the velocity to zero, ZUPT can be introduced as measurements into the Kalman Filter [33]. More importantly, ZUPT enables the Kalman Filter to correct most of the position drifts that occur during the stride phase [98, 28].

2.2.2.2 Pedestrian Dead Reckoning (PDR)

One of the simplest methods to track the moving body's trajectory is using PDR systems which are based on detecting the step events and moving the location forward by the step length in the direction that is determined for the user's orientation. Traditionally, researchers analyze the acceleration signals to detect peaks [39] or zero-crossing events [7, 8] to count the number of steps. Recently, learning-based approaches have been utilized for step-counting purposes. The phone's orientation can be obtained by integrating the data from the accelerometer and gyro. In [31], a system is proposed based on dynamic programming to estimate the discrete walker's orientation along with drift. Although the algorithm performed well in tests with sighted walkers, the quality of the results degraded in tests with blind walkers. The main challenge is that the orientation of the phone needs to be decoupled from the direction of walking, for example, if one at some point re-positions the phone on their body. Step length estimation (SLE) plays a crucial role in the operation of PDR systems. SLE is a challenging problem as stride length can vary with gender, height, weight, and health condition [110]. Traditional empirical methods use predefined coefficients and acceleration values to estimate the step length [111, 54, 61]. Recently, learning-based methods have been employed for step length estimation problems [40, 110, 55, 116].

2.3 Data-Driven Odometry

Recent advancements in inertial odometry have introduced learning-based methods. In the earliest work, RIDI (Robust IMY Double Integration) regresses velocity vector from angular velocity and linear acceleration data to optimize bias but still relies on double integration from the corrected IMU data for position estimation [117]. IONet takes a different approach by feeding accelerometer and gyroscope measurements into a deep neural network. This method estimates the user's heading and regresses the change in distance in a predefined time window [14]. RoNIN stands out as it processes inertial data in a heading-agnostic reference frame, utilizing three distinct neural network architectures: an LSTM network, a temporal convolutional network (TCN), and a residual network (ResNet). An impressive feature of RoNIN is its capacity to function seamlessly even when the phone is repositioned on the user's body. This is achieved by decoupling the phone's orientation from the user's orientation, ensuring effective performance, even if the user shifts the phone to a different pocket during walking [45]. In recent work, TLIO (Tight Learned Inertial Odometry) introduces regression for 3D displacement estimates using a ResNet architecture. Subsequently, by utilizing the raw IMU measurements and employing the estimated states as measurements through a stochastic cloning Extended Kalman Filter (EKF), the method allows for the extraction of sensor biases, position, and orientation [66].

2.4 Inertial Navigation for Blind Users

Performing everyday tasks, such as navigating their surroundings, poses a significant challenge for visually impaired individuals. Consequently, there has been a growing focus on developing assistive technologies to facilitate independent travel for the blind. Particularly, indoor navigation presents additional complexities, as GPS signals are often limited in such environments, such as office buildings, shopping malls, and transit hubs [74, 82]. Assistive navigation systems incorporate various technologies, including location-based information to navigate obstacles (e.g., announcing the presence of nearby stairs) and wayfinding systems that provide directions or assist users in retracing their path to the starting point. In [35], visual odometry is employed for wayfinding. This approach uses computer vision to detect standard signs, such as exit signs, as beacons. Additionally, 2D maps are utilized to identify impassable barriers, leveraging particle filters to estimate and track the user's location in the environment. BLE beacon-based turn-by-turn navigation system is proposed in [3], which requires the availability of the environment map. In another study [25], authors introduce a particle filter-based method that exploits inertial sensors and leverages visually impaired users' unique tactile sensing capabilities to confirm the presence of anticipated tactile landmarks along the provided path. Furthermore, in [29], an inertial sensor-based system is proposed for assisted return, utilizing a turns/steps representation of indoor environments.

2.5 Inertial Data Sets

Training learning-based approaches for odometry and assessing their performance rely on well-calibrated inertial data sets. Recent attention has been given to creating smartphone-based inertial data sets for indoor localization, considering the lack of publicly available data sets. Examples include the OxIOD data set, collected with pedestrians walking inside a room using a smartphone at different placements, recording IMU data. Additionally, an optical motion capture system (Vicon) provides high-precision labels with locations, velocities, and orientations of the user at each time [15, 14]. The RIDI data set comprises IMU sensor measurements and 3D motion trajectories obtained from the Visual Simultaneous Localization and Mapping (VSLAM) system on a smartphone equipped with an RGBD camera, specifically the Google Tango phone [117]. RoNIN collected a diverse and large-scale data set using two devices: one for recording IMU measurements and another (Asus Zenfone AR) for ground-truth motion trajectories [45]. In the data set presented in [110], a smartphone provides inertial data, and a foot-mounted sensor generate ground truth for heel strike moments and stride length.

2.5.1 WeAllWalk Data Set

The gait patterns of sighted individuals may differ from those of blind walkers, emphasizing the importance of using data sets that include blind travelers for developing odometry systems for the visually impaired. All previously mentioned data sets are

limited to data collected from sighted individuals. The WeAllWalk data set stands out as the sole publicly available collection of inertial data from blind individuals. In this study, ten blind volunteers and five sighted walkers participated, with nine blind walkers using a long cane, one using a dog guide, and two alternating between a long cane and a dog guide. Collectively, participants covered 7 miles of long and complex routes, segmented into "straight" or "turn" categories on floor maps, with corresponding coordinate information provided. Two iPhones (6s) were carried by participants at different locations to record inertial and attitude data (sampled at 25 Hz), and two foot-mounted sensors on participants' shoes generated ground truth labels for heel strike times. Additionally, the data set records the time each walker crossed segment boundaries (waypoints) and timestamps specific events such as door openings, collisions with obstacles, or getting caught in a door opening, which are tagged as features [30].

Chapter 3

Step Counting

3.1 Introduction

To create an effective wayfinding system for the visually impaired, a dependable inertial odometry system is crucial. Wayfinding strategies differ when a building map isn't available. In such cases, the navigation system can be useful in assisted return by providing directional guidance to visually impaired user who is backtracking their path. Whereas, when a map is accessible, the routing information can be provided by the system to the selected destination. Fundamental to tracking the moving body is step detection, altering the position according to the user's step length and direction. Hence, step counting serves as the core of pedestrian dead reckoning (PDR) systems, drawing considerable attention in research and commercial applications.

Commercial pedometers commonly employ sensors embedded in shoes, attached to ankles, or belts. Additionally, step counting is integrated into contemporary smart-

watches and smartphone apps for fitness monitoring. Traditionally, step counting involves identifying peaks or features within acceleration or rotation rate signals [5, 51]. Inspired by previous work [23], I developed a recurrent neural network (RNN)-based step counting system. User heading direction can be obtained from azimuth angles provided by modern smartphone operating systems or derived from a turn-detection module (developed by another researcher in our group) for users moving along paths with discrete turning angles. Various methods for estimating step length will be discussed in the next chapter. This chapter introduces a step counter model, presenting experimental results achieved with the WeAllWalk data set and displaying the results obtained through a PDR system that includes the step counter as a key component.

3.2 Step Counter Model

We've developed a step-counting system based on LSTM (Long Short-Term Memory) [46], a popular type of Recurrent Neural Network (RNN). Past research [23] utilized a bi-directional LSTM, which can enhance robustness by considering a whole batch of data simultaneously. However, this approach might not be suitable for scenarios necessitating timely step detection, such as wayfinding or assisted return applications where continuous tracking of the walker's position along the route is critical. Thus, we opted for a regular, uni-directional LSTM for our application.

We can obtain the inertial data of interest from the phones by utilizing the device's proprietary sensor fusion algorithm, which processes data collected through the

onboard accelerometers and gyroscopes. The desired data including attitude, representing the 3-D orientation relative to a fixed "world" reference frame where the Z axis aligns with gravity; gyro, providing a 3-D vector of angular velocities; and user acceleration, a 3-D vector indicating the actual acceleration of the phone (i.e., excluding the force of gravity).

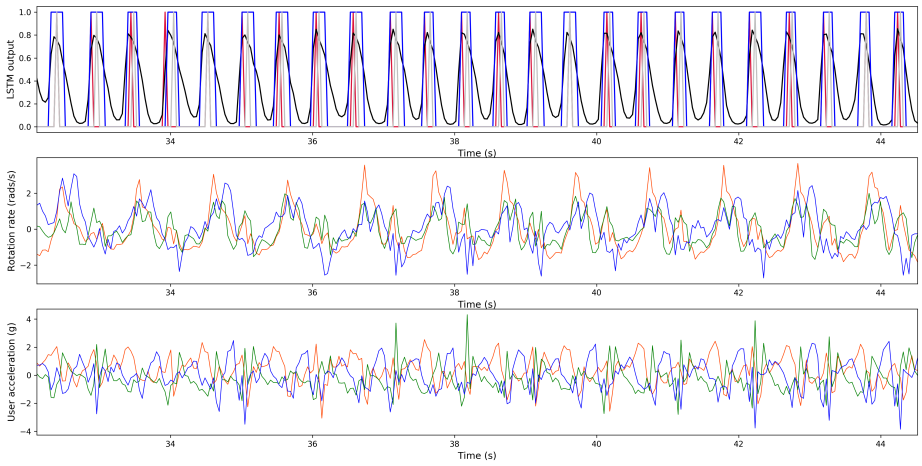
Our LSTM takes as input the user acceleration (across three axes) and rotation rate (along three axes). These vectors are pre-multiplied by the inverse of the attitude matrix to establish a reference frame independent of heading [45]. Each axis's data is normalized to zero mean and unit variance. The LSTM is trained to generate a sequence that approximates the desired output, a sequence uniformly set to 0, except during heel strikes, where it is set to 1. More precisely, we convert each impulse into a narrow triangular wave spanning three samples. The LSTM is trained using Keras with 100-sample windows (equivalent to 4 seconds) and utilizes the least squares loss function. Note that the output of LSTM is a sequence of numbers between 0 and 1, and is transformed to a binary signal by applying a suitable threshold S. Within each series of consecutive LSTM output samples surpassing this threshold, we designate the midpoint as the estimated time of heel strike.

Our LSTM consists of a 2-layer network with a hidden unit size of six. Through initial experiments, we found this network depth to be sufficient for the task, as adding more layers increased the risk of overfitting. We employed the Adam optimizer and dropout for regularization. Training occurs with a batch size of 256 and an initial learning rate of 0.005, reduced by half after 50 epochs, across a total of 64 epochs.

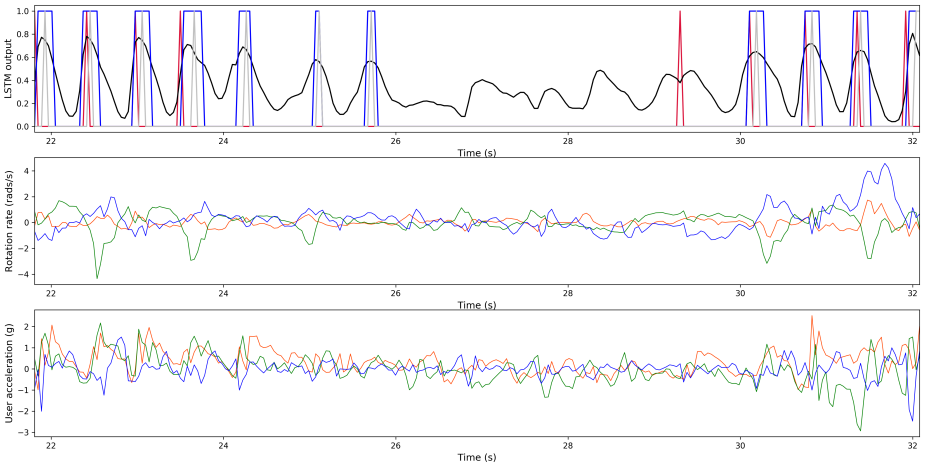
Figure 3.1 illustrates instances of step detection. While our step counter performs reliably in most conditions, we observed a decaying oscillatory pattern in the LSTM output when a person stops walking. Additionally, occasional misses of the initial one or two steps might occur when starting to walk.

3.3 Step Length

To apply step counting in odometry, defining the walker's step length is crucial. Various techniques, including learning-based methods using inertial data, have been proposed to estimate step length. In my experiments with the WeAllWalk data set, I tested several of these algorithms but encountered challenges in achieving consistent and accurate results. The primary issue arises from the inability to establish ground truth step length labels within this data set. Consequently, I derived a fixed step length based on the known lengths of paths traversed in WeAllWalk, and the ground-truth number of steps recorded for each path by each participant within the training set.



(a)



(b)

Figure 3.1: Examples of step detection. (a) Top row: the output of LSTM (black line) is thresholded, and the midpoints of the resulting positive segments (gray line) are taken as the estimated times of heel strike (ground-truth shown by red line). The LSTM takes in input the 3-axes rotation rate (middle row) and the 3-axes user acceleration (bottom row). In (b), instances of overcounting are observed between $t = 24$s and $t = 26$s, while an instance of undercounting is visible between $t = 29$s and $t = 30$s

3.4 Experiments

3.4.1 Training/Test Modalities

Considering the distinct walking characteristics between long cane users and guide dog users [47], we analyzed these groups separately, denoted by the modifiers :LC (for long cane users) and :GD (for guide dog users). We employed the following training/test schemes, ensuring that the system testing a specific walker was never trained using data from that same individual.

- **Train on Sighted (TS):** All parameters are derived using data gathered from the five sighted walkers within WeAllWalk. The system's performance is then evaluated on the two communities of blind users (TS:LC, TS:GD). This scenario simulates the use of a system initially designed for sighted walkers when employed by blind walkers without specific tailoring or customization.

- **Train in same Community (TC):** In this setup, the system, tested with long cane users, guide dog users, and sighted individuals, was trained exclusively on data from walkers within the same community. We utilized the Leave-One-Person-Out (LOPO) cross-validation strategy [37, 57], wherein each participant was tested using a system trained on data from all other participants within the same community. It's important to note that each training set in the TC:GD modality comprises data from only two walkers.

- **Train on All (TA):** In this configuration, all available data in WeAllWalk was

utilized for training, employing the Leave-One-Person-Out cross-validation strategy (TA:LC, TA:GD). For instance, a long cane user is tested with a system trained on data including all sighted participants, guide dog users, and other long cane users.

For all tests in each modality, the measured quantities of interest are averaged over both iPhones carried by the participants, all paths, and all participants in the test set.

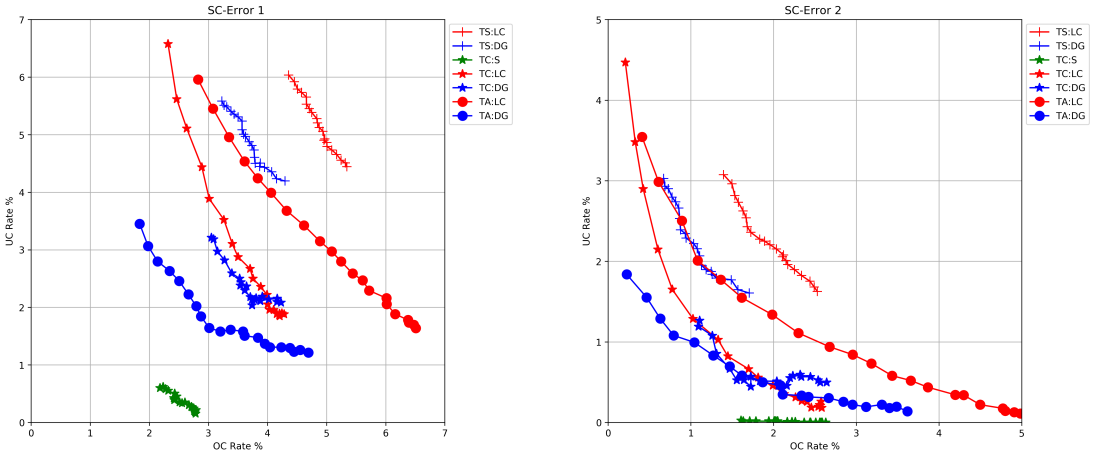
3.4.2 Step Counter Results

We considered two different error metrics for the step counter:

- **SC-Error 1:** We compute the midpoint between any two consecutive ground-truth heel strike times, and count the number of detected steps between two consecutive such midpoints. Note that there is exactly one ground-truth heel strike within this interval. If n>1 steps are detected within this interval, n-1 overcount events are recorded. If no steps are detected, an undercount event is recorded.

- **SC-Error 2:** The difference between the number of detected steps in a WeAllWalk segment and the number of ground-truth heel strikes in the same segment is recorded as overcount (if positive) or undercount (if negative).

The total number of overcount and undercount events in each path is normalized by the total number of ground-truth heel strikes to produce an undercount (UC) rate and an overcount (OC) rate. Note that increasing the threshold S on the output of the LSTM normally results in an increase in the UC rate and a decrease in the OC rate. This is shown in Figure 3.2, where we plotted the UC rate vs OC rate as a function of S.



(a) (b)

Figure 3.2: UC rate vs OC rate curves as a function of the threshold S on the LSTM output.

The test results are presented in Table 3.1. The threshold S is determined to balance the OC and UC rates of SC-Error 2 measured in the training data. Thus, these S values are computed and averaged over all cross-validation rounds (refer to Table 3.1). Additionally, the average step length (SL) obtained from the training data is also included in Table 3.1.

	SC-Error 1		SC-Error 2		Mean S	Mean SL (m)
	UC rate %	OC rate %	UC rate %	OC rate %		
TS:LC	10.58	2.51	8.20	0.13	0.78	0.74
TS:GD	8.74	2.08	6.75	0.1	0.78	0.74
TC:S	3.03	1.05	2.24	0.26	0.78	0.74
TC:LC	**3.22**	**3.27**	**0.91**	**0.96**	0.55	0.55
TC:GD	4.99	2.15	3.23	0.39	0.68	0.62
TA:LC	4.06	3.8	1.42	1.17	0.56	0.62
TA:GD	**2.29**	**2.55**	**0.83**	**1.08**	0.55	0.62

Table 3.1: Undercount (UC) and overcount (OC) rates of step counter, along with the mean threshold S and step length SL. Boldface highlights the pair (UC rate, OC rate) with the smallest sum for each community of blind walkers (LC, GD).

The results presented in Table 3.1 and the curves illustrated in Figure 3.2 clearly show how the choice of the walker community used for training the system significantly impacts the accuracy of step counting. For instance, when evaluating with long cane users, using a step counter trained on sighted walkers (TS:LC), the sum of undercount and overcount rates were 13.09% (SC-Error 1) or 8.33% (SC-Error 2). However, by training the system solely on data from other long cane users (TC:LC), these numbers decreased remarkably to 6.49% and 1.87%, respectively. The best outcomes for guide dog users were achieved when training on entirety of available data (TA:GD). The limited number of guide dog users explains why TC:GD doesn't surpass TA:GD.

Prior studies applied various step counting algorithms to WeAllWalk in [30]. Our findings indicate a notable improvement with the utilization of an LSTM model.

For example, the lowest SC-Error 1 value (measured as the sum of UC and OC rates) for long cane users was found to be 7.8% in [30] compared to 6.5% with our system. Similarly, for the same user community, the minimum SC-Error 2 reported in [30] was 4.8%, while our system achieved 1.9% (refer to Table 3.1).

The mean threshold S derived from the LSTM output notably differs across walker communities, with sighted walkers exhibiting a substantially larger average value (0.78) compared to guide dog users (0.68) and long cane users (0.55). Larger thresholds should be expected when the output of the LSTM is closer to the binary signal used to indicate heel strikes. This suggests that the LSTM is better capable of modeling the desired output for sighted walkers (possibly due to their more regular gait) than for blind walkers.

The average step lengths learned within each community also exhibit variations, with sighted walkers demonstrating a larger average step length (0.74 m) compared to guide dog users (0.62 m) and long cane users (0.55 m). This aligns with expectations as guide dog users typically walk at a faster pace than long cane users, as they do not need to probe the space ahead with the cane and can rely on their guide dog.

3.4.3 PDR System Results Built by Step-Counter Module

A Pedestrian Dead Reckoning (PDR) system was constructed by integrating the step counter module with a heading tracker, employing either the azimuth angle or a turn detector module (developed by another researcher in our group), and a fixed step length. This PDR system effectively reconstructs a user's trajectory, as evaluated on the

WeAllWalk data set, under the assumption of an unavailable building map. However, with the availability of a map, further enhancements are feasible. Integration of particle filtering with a mode searching mechanism, leveraging spatial information from the map (implemented by another researcher in our group), could significantly augment the designed system's wayfinding capabilities.

The path reconstructed by the PDR system was assessed against RoNIN [45], a state-of-the-art deep learning-based odometry system optimized for data set collected from sighted individuals. RoNIN processes inertial data in a heading-agnostic reference frame, utilizing acceleration, raw gyro data, and attitude data as inputs to compute the walker's velocity, which is then integrated over time to determine the location.

Adapting RoNIN for use on the WeAllWalk data set required up-sampling the required inertial data (via a linear interpolation method) from its original 25 Hz acquisition rate to 200 Hz, matching RoNIN's training data rate. Utilizing the authors' open-source implementations of RoNIN (ResNet18 architecture)[1], I conducted tests on the WeAllWalk data set. As anticipated, results might vary due to discrepancies between the sensors used in RoNIN's training and those in the iPhones utilized for WeAllWalk. To mitigate this, a basic adjustment was made by computing a scaling factor through least squares regression. This factor, found to be 1.27, aimed to align the magnitudes of velocities generated by RoNIN with the ground-truth velocities in WeAllWalk. This scaling factor was subsequently applied to the velocity vectors produced by RoNIN. Additionally, another researcher in our group fine-tuned RoNIN using data obtained from

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

blind walkers within the WeAllWalk data set.

3.4.3.1 Path reconstruction

The WeAllWalk data set only provides the timestamps $\{t_j^i\}$ indicating when a walker reached each waypoint. Therfore, to establish a ground truth, a basic assumption was made that each user traversed the middle of the corridor width during transitions between waypoints. Additionally, due to the lack of defined reference frame for the estimated trajectory, alignment with the ground-truth path was necessary (implemented by another Ph.D. researcher in our group). Following this alignment, the evaluation of the estimated trajectories involves three key metrics.

- The first metric considered is the RMSE of estimated waypoint locations.

$$\text{RMSE}_{wp} = \sqrt{\frac{1}{N}\sum_j ||\bar{P}_j^i - P^i(t_j^i)||^2}$$

  where N is the number of waypoints in the path, $\{\bar{P}_j^i\}$ represents the ground truth locations of the waypoints, and $\{P^i(t_j^i)\}$ are the estimated waypoints locations derived by waypoints timestamps.

For the remaining two metrics, I sampled the estimated trajectory into N_e^i points $\{Q_m^i\}$, and the ground truth path into N_{gt}^i points $\{\bar{Q}_n^i\}$ with a uniform inter-sample distance of 1 m. Given these two sets of points, the reconstructed path can be assessed by the following two metrics.

- The second metric is Hausdorff distance [92] between the two sets of points.

$$\text{Haus} = \max\left(\max_m\left(\min_n\left(||Q_m^i - \bar{Q}_n^i||\right)\right), \max_n\left(\min_m\left(||Q_m^i - \bar{Q}_n^i||\right)\right)\right)$$

- The third metric is the average Hausdorff between the two sets of points.

$$\text{avHaus} = \frac{1}{2}\left(\sqrt{\frac{1}{N_e}\sum_{m}\min_{n}\left(||Q_m^i - \bar{Q}_n^i||^2\right)} + \sqrt{\frac{1}{N_{gt}}\sum_{n}\min_{m}\left(||Q_m^i - \bar{Q}_n^i||^2\right)}\right)$$

The Hausdorff distance highlights significant differences, even if they occur intermittently, between the estimated and ground truth trajectories. On the other hand, the average Hausdorff distance, detecting consistent biases. Together, these metrics provide a comprehensive assessment of the estimated trajectory's overall accuracy, beyond just the waypoints. Note that these evaluation metrics (except the sampling algorithm) are implemented and calculated by another Ph.D. researcher in our group.

The following algorithms are considered for map-less path reconstruction (Figure 3.3).

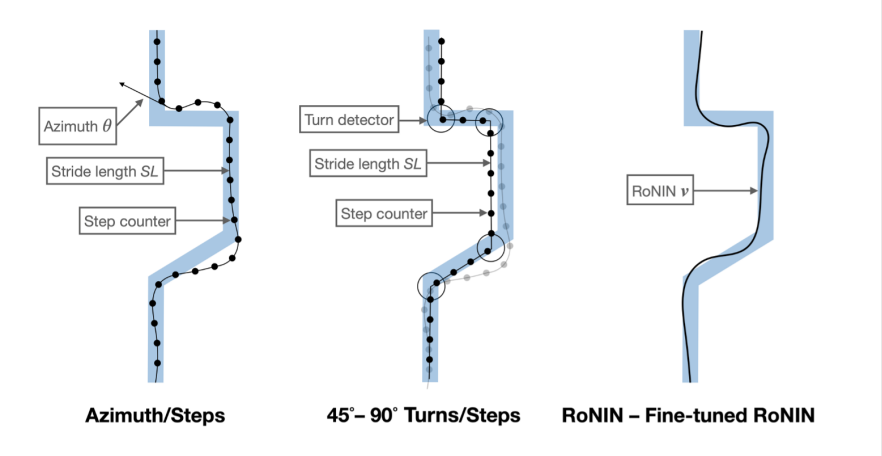


Figure 3.3: Three algorithms used for path reconstruction. The blue line represents the path taken by the walker. The black line represents the estimated path. Dots represent heel strikes; and circles represent turns.

- **Azimuth/Steps (A/S):** With every detected step, a displacement vector is defined, its length set to the fixed step length, and its orientation equal to the azimuth angle θ as provided by the phone.

- **45°-90° Turns/Steps (T/S):** This algorithm is similar to the first one, but uses the orientation from a two-stage 45° or 90° turn detection module (implemented by another Ph.D. student in our group) under the assumption that users navigate through corridors forming a network intersecting at these angles.

- **RoNIN (R) – Fine-tuned RoNIN (FR):** The trajectory is reconstructed by applying either RoNIN or Fine-tuned version of RoNIN (developed by another Ph.D. student in our group) on the WeAllWalk data set, which estimates position by integrating velocity outputs over time.

The Table 3.2 reports reconstruction errors across three metrics for the considered algorithms. It's important to note that the assessment involving fine-tuned RoNIN was exclusively conducted for blind walkers (LC and GD). The smallest reconstruction error for all metrics are obtained by 90°Turns/Steps algorithm.

	A/S			45° T/S			90° T/S			R			FR		
TS:LC	9.43	14.68	4.90	9.17	14.48	4.7	9.03	13.90	4.63	5.43	8.56	2.93	–	–	–
TS:GD	5.81	8.97	3.30	5.64	8.67	2.82	4.94	**7.50**	2.75	5.66	8.55	2.84	–	–	–
TC:S	4.45	7.06	2.39	3.96	6.12	1.89	**3.93**	**5.97**	**1.88**	4.26	6.75	2.39	–	–	–
TC:LC	3.85	6.21	2.30	3.86	6.45	2.22	**3.46**	**5.47**	**1.97**	5.43	8.56	2.93	4.36	7.37	2.54
TC:GD	6.38	9.90	3.29	6.28	9.86	2.92	6.13	9.60	2.92	5.66	8.55	2.84	6.80	10.42	3.29
TA:LC	6.29	10.27	3.60	5.99	9.79	3.32	5.88	9.47	3.31	5.43	8.56	2.93	6.18	9.90	3.27
TA:GD	5.21	8.37	2.88	5.00	8.18	2.52	**4.59**	7.64	**2.50**	5.66	8.55	2.84	5.17	8.08	**2.50**

Table 3.2: Reconstruction errors ($RMSE_{wp}$, Haus, avHaus) for path reconstruction algorithms. Boldface indicates the smallest error values for each metric across walker communities (S, LC, GD).

3.5 Conclusions

In this chapter, we introduced a step counter module, a fundamental element within any Pedestrian Dead Reckoning (PDR) system. Utilizing a fixed step length, and a heading tracker (implemented by another Ph.D. researcher in our group), we constructed and assessed a PDR system on the WeAllWalk data set. Additionally, we conducted a comparative analysis against RoNIN, a state-of-the-art odometry system trained by sighted users. The obtained results emphasize the significance of the walker community selected for training the algorithm's parameters in our study. Systems trained solely with sighted walkers yielded consistently poorer performance when evaluated with long

cane users and, to a lesser degree, with guide dog users. However, incorporating training data from these specific communities notably enhanced performance, even comparable to the best outcomes achieved when tested with sighted walkers (refer to Table 3.2). Furthermore, our findings indicate superior performance of the Turns/Steps PDR system compared to the learning-based model, RoNIN (see Table 3.2).

Chapter 4

Step Length

4.1 Introduction

In order to use step counting for odometry, it is essential to determine the walker's step length. The accurate estimation of step length is a critical component of PDR systems, and it can be quite challenging due to the variations in gait patterns influenced by factors such as gender, height, weight, age, and health condition [56]. Traditional methods rely on predefined coefficients and acceleration values for step length estimation. For instance, the Weinberg method exploits the relationship between the step length and the difference of max and min values of the vertical acceleration within the step [111]. Kim et al. [54] compute step lengths based on the average of the acceleration norm, while Ladetto [61] utilizes the local variance of the acceleration signal. These methods typically involve a calibration process to determine the system coefficients. Linear models have been proposed in studies like [19, 83], where step length is calculated

based on a combination of the user's step frequency and height. However, a significant limitation of these methods is their reliance on prior knowledge of the user's height.

Recently, learning-based methods have been introduced to address the step length estimation problem. In the work presented in [40], an autoencode-based model is employed. This model learns valuable features from the accelerometer and gyroscope data through stacked autoencoders using a greedy layer-wise training approach. The final regression layer is then used to estimate step length using the learned features. The same approach is employed in [116], though deep belief network (DBN) is used instead of autoencoders for feature learning. StepNet [55] combines a traditional method with a learning-based model. In this scheme, the higher-level features extracted from raw inertial data along with the smartphone location on the body (e.g., pocket, swing, texting) are plugged to a convolutional neural network (CNN)-based model to estimate the Weinberg gain coefficient and subsequently calculate step length. In the study described in [110], step lengths are predicted using a combination of Long Short-Term Memory (LSTM) and an autoencoder model. The LSTM network first extracts temporal dependencies and features from the raw inertial data. These learned features, along with traditional features, are then fed into an autoencoder to train a noise-robust encoder. Finally, a regression layer is applied to predict step length.

As mentioned in the previous chapter, we conducted experiments with multiple algorithms; however, achieving consistent and accurate results on the WeAllWalk data set proved to be challenging. Many of these learning-based approaches rely on ground truth data created using foot sensors. During our investigation, we discovered that the

accelerometer range of the foot sensors in the WeAllWalk data set was limited to $\pm 2g$. Unfortunately, this limitation prevented us from creating the necessary ground truth data using those sensors. Consequently, we were required to gather a new data set to address this limitation

4.2 Data Collection

In this section, we outline our methodology for collecting a representative data set, and obtaining precise "ground truth" annotations. Our study was designed to capture inertial data covering a wide range of stride lengths, each of which was accurately measured. Note that the common practices (e.g., [55, 91, 40]) such as measuring step lengths by dividing the path length by the number of steps taken in that path, can yield inaccurate results. This is particularly relevant because, during "natural" walking, an individual's step length exhibits noticeable variations, with the standard deviation often ranging from 3% to 7% of the average, as measured in previous studies [91]).

Twelve participants (6 female and 6 male, average age: 35.6) walked through an office building, completing four paths, except for participant P10, who walked three paths. Each path consisted of multiple sub-paths that began and ended at marked locations (see Figure 4.1). For each sub-path, one of three possible categories of stride length was prescribed: "natural", "shorter than natural", or "longer than natural". However, no other directions were given to the participants, allowing them to choose their own pace for the prescribed stride length category. The sub-paths were chosen such that the total

distance covered was consistent across all stride length categories (equal to 236 meters, for a total traversal length of 709 meters per participant.) The number of steps taken by the participants ranged from 789 (P10) to 1344 (P9).

(a) Path 1 and 2

(b) Path 3 and 4

Figure 4.1: Four paths taken by users on the 3rd floor of the engineering building at UCSC. (a) Paths 1 and 2 are shown in red and dashed blue lines, respectively, with a total length of 111.73 meters. (b) Path 3, totaling 254.27 meters, is represented by dashed orange lines, and the segments taken are numbered. Path 4, shown in purple lines, covers a distance of 231.19 meters.

Each participant carried two inertial IMU packages (Xsens DOT), with each package tied to one shoe using an elastic band. The IMUs produced data from 3 accelerometers and 3 gyros at the rate of 120 samples/s with 16 bits resolution. The accelerometers and gyros were configured to measure data over a scale of ± 16 g (accelerometers) and

$\pm 2,000°/s$ (gyroscopes). Additionally, each participant had two smartphones, an iPhone 13 Pro and an iPhone X. One smartphone was placed in the back pocket of their pants, while the other was held at chest height as if the participant were looking at its screen. These smartphones ran an application that collected time-stamped inertial data at a rate of 120 samples per second. Both the IMU packages and the smartphones were synchronized to a common time scale.

4.2.1 Stride length by foot-mounted sensors

First, following [28], we computed and removed the gyroscope bias by averaging the measurements during the stationary time at the beginning and end of each path. We then processed the inertial data recorded from each foot sensor using and EKF-based dead-reckoning algorithm with Zero-velocity UPdaTes (ZUPT) and HDR (Heading Drift Reduction) corrections to obtain accurate measurements of stride length, defined as the path traversed between two consecutive heel strikes of the same foot.

4.2.1.1 INS process

Based on prior experiments and using an ablation study, we verified that our EKF odometry algorithm, which tracks sensor biases along with estimated attitude errors, velocity errors, and position errors in its state vector, provides the best results. The 15-element error state at time k:

$$\delta x_k = [\delta\varphi_k,\, \delta\omega_k^b,\, \delta r_k,\, \delta v_k,\, \delta a_k^b]$$

The algorithm pipeline, consisting of six steps, is shown in Figure 4.2. First, the estimated sensor biases by EKF are removed from inertial data. Subsequently, the bias-compensated angular rate is utilized to update the rotation matrix using the *Pade'* approximation.

$$C^n_{b_{k+1|k}} = C^n_{b_{k|k}} \frac{2I_{3\times3} + \delta\Omega_{k+1}\Delta t}{2I_{3\times3} - \delta\Omega_{k+1}\Delta t}$$

where $\delta\Omega_{k+1}$ is the skew-symmetric matrix for angular rates used to define small angular increments in orientation.

$$\delta\Omega_{k+1} = \begin{bmatrix} 0 & -\omega^{b'}_{z_{k+1}} & \omega^{b'}_{y_{k+1}} \\ \omega^{b'}_{z_{k+1}} & 0 & -\omega^{b'}_{x_{k+1}} \\ -\omega^{b'}_{y_{k+1}} & \omega^{b'}_{x_{k+1}} & 0 \end{bmatrix}$$

As a third step, the bias compensated acceleration measurements are rotated to the navigation frame, and the gravity component is removed.

$$a^n_{k+1} = C^n_{b_{k+1|k}} \cdot a^{b'}_{k+1} - \begin{pmatrix} 0 & 0 & g \end{pmatrix}^T$$

Next, the acceleration will be integrated to obtain the velocity, and the velocity will be integrated to obtain the position in the navigation frame. At the fifth step, the velocity and position corrections will be applied based on the EKF error estimates. Finally, the EKF attitude error estimates ($\delta\varphi_k$) will be utilized to apply the attitude correction.

$$C^n_{b_{k+1|k+1}} = \frac{2I_{3\times3} + \delta\Theta_{k+1}\Delta t}{2I_{3\times3} - \delta\Theta_{k+1}\Delta t} C^n_{b_{k+1|k}}$$

where $\delta\Theta_{k+1}$ is the skew symmetric matrix for small angles.

$$\delta\Theta_{k+1} = \begin{bmatrix} 0 & -\delta\varphi_k(3) & \delta\varphi_k(2) \\ \delta\varphi_k(3) & 0 & -\delta\varphi_k(1) \\ -\delta\varphi_k(2) & \delta\varphi_k(1) & 0 \end{bmatrix}$$

After each measurement update, the EKF transfers the error states to the INS and resets the error state vector to zero since those errors are already compensated and integrated into the INS estimations. Consequently, steps 4 and 6 are only essential after the EKF measurement updates are applied.

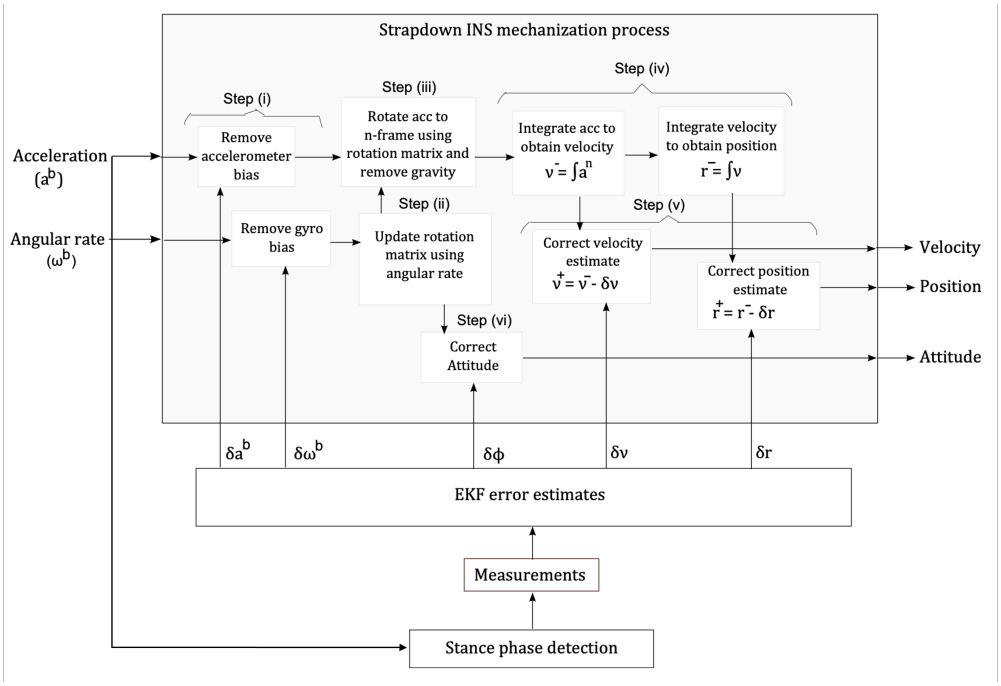


Figure 4.2: EKF-based inertial navigation system [98].

4.2.1.2 EKF measurements

ZUPT is a common technique used to enhance the accuracy of INS and is typically integrated into the INS system through a Kalman filter. The assumption that velocity should be close to zero when the foot is nearly static on the ground is the basis for ZUPT. As a result, the difference between the INS-calculated velocity and the zero-velocity measurement is introduced into the Extended Kalman Filter (EKF) as a ZUPT measurement. This technique is highly effective in reducing errors in velocity estimates, particularly during periods of rest or when there is no motion.

$$Z_{ZUPT} = v_{k|k+1} = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T - \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$$

In addition, HDR (Heuristic Heading Drift Reduction) is proposed to mitigate heading drift by leveraging the observation that many walking paths in various applications are straight. Generally, when the user is walking along a straight path, the yaw angle typically experiences minimal variation. Therefore, straight walks can be detected by analyzing the changes in the yaw angle. If the fluctuations in the yaw angle among consecutive steps remain below a certain threshold, it indicates that the user is walking in a straight direction. In such cases, the variations in the yaw angle can be used as an additional measurement input to the EKF for more accurate heading estimation [52].

$$Z_{HDR} = \begin{cases} \Delta\psi_t & \text{if } |\Delta\psi_t| < th_{\Delta\psi} \\ 0 & \text{otherwise} \end{cases}$$

4.2.1.3 EKF error estimates

The Kalman filter comprises two main steps: the prediction step and the update (or correction) step. The error covariance gets predicted at each time sample, but it only gets corrected when the stance phase is detected and measurements become available.

$$P_{k+1|k} = \Phi_{k+1} P_{k|k} \Phi_{k+1}^T + Q$$

where Φ_{k+1} is the state transition matrix and Q is the process noise covariance matrix.

$$\Phi_{k+1} = \begin{bmatrix} I_{3\times3} & \Delta t C^n_{b_{k+1|k}} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & \Delta t I_{3\times3} & 0_{3\times3} \\ -\Delta t S(a^n_{k+1}) & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & \Delta t C^n_{b_{k+1|k}} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix}$$

where $S(a^n_{k+1})$ is the skew-symmetric matrix formed from the accelerations in the navigation frame.

$$S(a^n_{k+1}) = \begin{bmatrix} 0 & -a^n_{z_{k+1}} & a^n_{y_{k+1}} \\ a^n_{z_{k+1}} & 0 & -a^n_{x_{k+1}} \\ -a^n_{y_{k+1}} & a^n_{x_{k+1}} & 0 \end{bmatrix}$$

The term $S(a^n_{k+1})$ relates the variation in velocity errors in the navigation frame to the variation in orientation errors. This relationship allows EKF to establish appropriate correlations within the error covariance matrix (P matrix). It's worth noting that there is no need to apply the error prediction step, $\delta x_{k+1|k} = \Phi_{k+1} \delta x_k$, within the EKF because

the EKF transfers the error states to the INS and resets the error state vector to zero after each measurement update.

When the EKF measurements (ZUPT and HDR) are available, first, kalman gain gets updated.

$$K = P_{k+1|k}H^T(HP_{k+1|k}H^T + R)^{-1}$$

where R is the measurement noise covariance matrix, and H is the measurement matrix.

$$H_{4\times15} = \begin{bmatrix} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{3\times3} & & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{bmatrix}$$

subsequently, the update step will be applied to both the state vector and error covariance matrix.

$$\delta x_{k+1} = KZ$$

$$Z = (Z_{HDR}, Z_{ZUPT})$$

$$P_{k+1|k+1} = (I - KH)P_{k+1|k}(I - KH)^T + KRK^T$$

4.2.1.4 Stance phase detection

Stance periods were identified based on three conditions as described in the following [52].

- Applying threshold on the magnitude of the acceleration measurements at any time k.

$$|a_k| = \sqrt{a_{xk}^2 + a_{yk}^2 + a_{zk}^2}$$

$$C_1 = \begin{cases} 1 & \text{if } th_1 \leq |a_k| \leq th_2 \\ 0 & \text{otherwise} \end{cases}$$

- The local acceleration variance, which represents the foot activity, must be below a given threshold.

$$\sigma_{ak}^2 = \frac{1}{2s+1} \sum_{j=k-s}^{k+s} (a_j - \bar{a}_k)^2$$

where s is the size of averaging window, and $\bar{a}_k$ is the local mean acceleration at time k, and it is computed by

$$\bar{a}_k = \frac{1}{2s+1} \sum_{q=k-s}^{k+s} a_q$$

Therefore,

$$C_2 = \begin{cases} 1 & \text{if } \sigma_{ak}^2 < th \\ 0 & \text{otherwise} \end{cases}$$

- Applying threshold on the magnitude of the angular rates.

$$|\omega_k| = \sqrt{\omega_{xk}^2 + \omega_{yk}^2 + \omega_{zk}^2}$$

$$C_3 = \begin{cases} 1 & \text{if } |\omega_k| < th_\omega \\ 0 & \text{otherwise} \end{cases}$$

When all these three conditions are met, the object is in the stance phase.

4.2.1.5 Prior experiments

In order to decide the different elements that we can track in the EKF state vector and different kinds of measurements that we can apply via EKF, we have implemented twelve variations of the EKF algorithm. These algorithms can be different in terms of the elements they track or the different measurements that they apply via EKF. We named the algorithms in a way that G and A represent that the gyroscope and accelerometer biases are included in the state vector. NoB stands for the case that the bias values are not included in the state vector. Other than ZUPT, and HDR, we have considered Zero Angular Rate Updates (ZARUs) as a measurement as well. During the stationary time when the foot is in contact with the ground, the angular rates, as well as the velocities, should theoretically be zero. Hence, ZARU can be applied along with ZUPT. To compare the algorithms, we calculated the Normalized Estimated Length (NEL) as an error metric. The total distance taken by a user is calculated by accumulating the step lengths and compared with the ground truth lengths of the paths. The error is then normalized by the ground truth length and averaged over all users and all paths. Figure 4.3 demonstrates the results achieved by different algorithms. The algorithm that tracks biases and applies HDR along with ZUPT gives us the best results (the last bar in the plot). A sample trajectory achieved by this algorithm with detected steps highlighted is demonstrated in Figure 4.4.

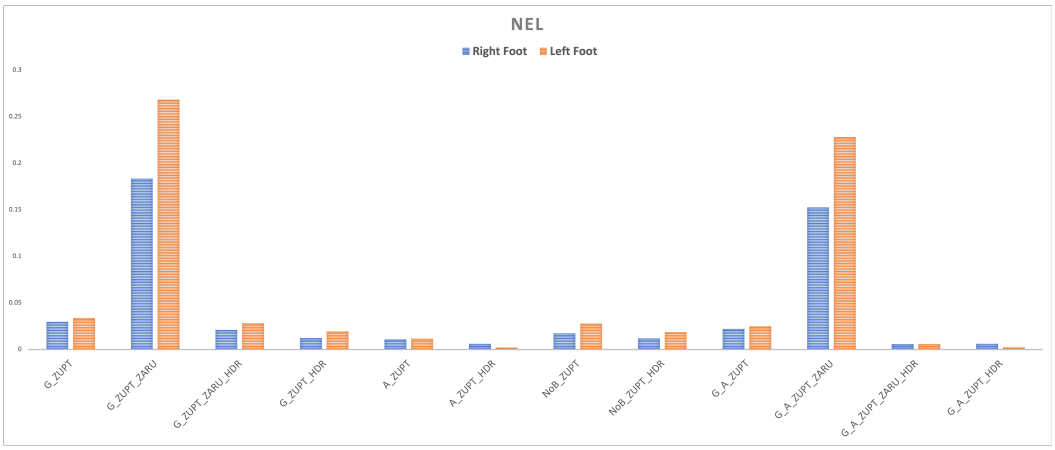


Figure 4.3: Results of an ablation study comparing different variations of the EKF algorithm based on the normalized estimated length error.

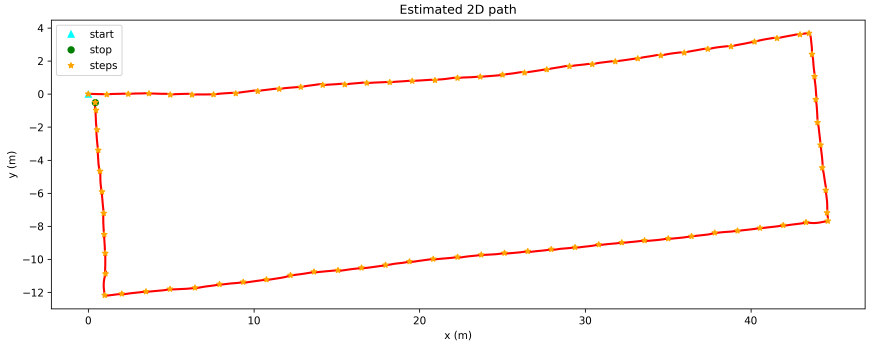


Figure 4.4: Reconstructed trajectory by best performed EKF algorithm, detected steps highlighted in yellow.

In prior experiments, we verified that this algorithm produced distance errors over long paths that were consistently less than 1% of the total traversed distance in the path. In order to measure stride lengths, we first detected each heel strike as the highest peak of the accelerometer magnitude within a window of 0.5 s around the beginning of

a stance period. Strides were measured by the path traversed between two consecutive heel strikes (shown as circles in Figure 4.5). Note that the ZUPT algorithm performs a "correction" of the estimated location during a stance period (see Figure 4.5).

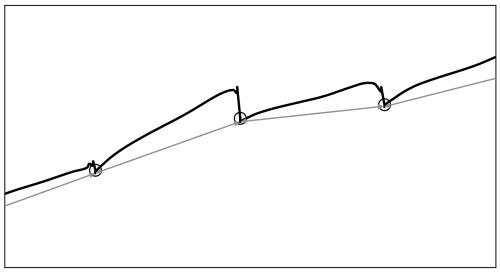


Figure 4.5: An example of a reconstructed trajectory for one foot-mounted sensor (black line). Note that the ZUPT algorithm applies a correction at each detected stance phase. The circles represent heel strike times, which are used to compute individual stride lengths (shown by grey arrows).

Figure 4.6 displays the distribution of stride lengths measured for all participants in our data collection (represented in purple) with a standard deviation of $\sigma = 0.38$m. The same figure illustrates the stride length distribution in the data set described in [110], which was also collected using a foot-mounted sensor (represented in green). It is worth noting that the prior data set was constructed from walkers who maintained a relatively uniform stride length with a standard deviation of $\sigma = 0.12$ m. We emphasize that a wide distribution of stride lengths is crucial for a thorough evaluation of an odometry algorithm.

While foot-mounted sensors are are well-suited for measuring stride lengths, when processing inertial data from a smartphone, it is more convenient to measure step

lengths, where a step is a path traversed by the user's body between two consecutive heel strikes from opposite feet. Note that the data recorded by the phone is approximately periodic over steps. Within a step, one of the two feet is in the stance phase, while the other is in the swing phase. To establish a ground truth step length, denoted as $\hat{l}_i$, we calculate the mean of the two overlapping stride lengths (one per foot) and divide the result by 2. Additionally, we measure the (average) walking speed during the same step as $\hat{v}_i = \hat{l}_i/\hat{T}_i$, where $\hat{T}_i$ represents the step period, defined as the time between two consecutive heel strikes from opposite feet.

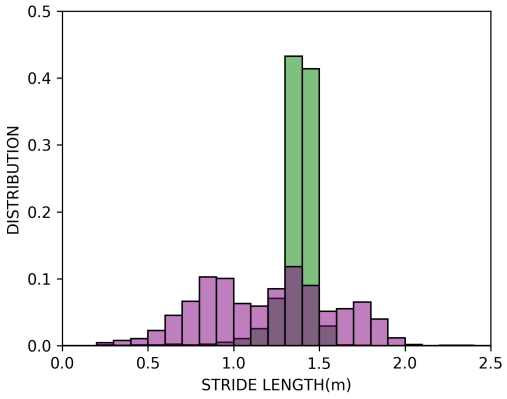
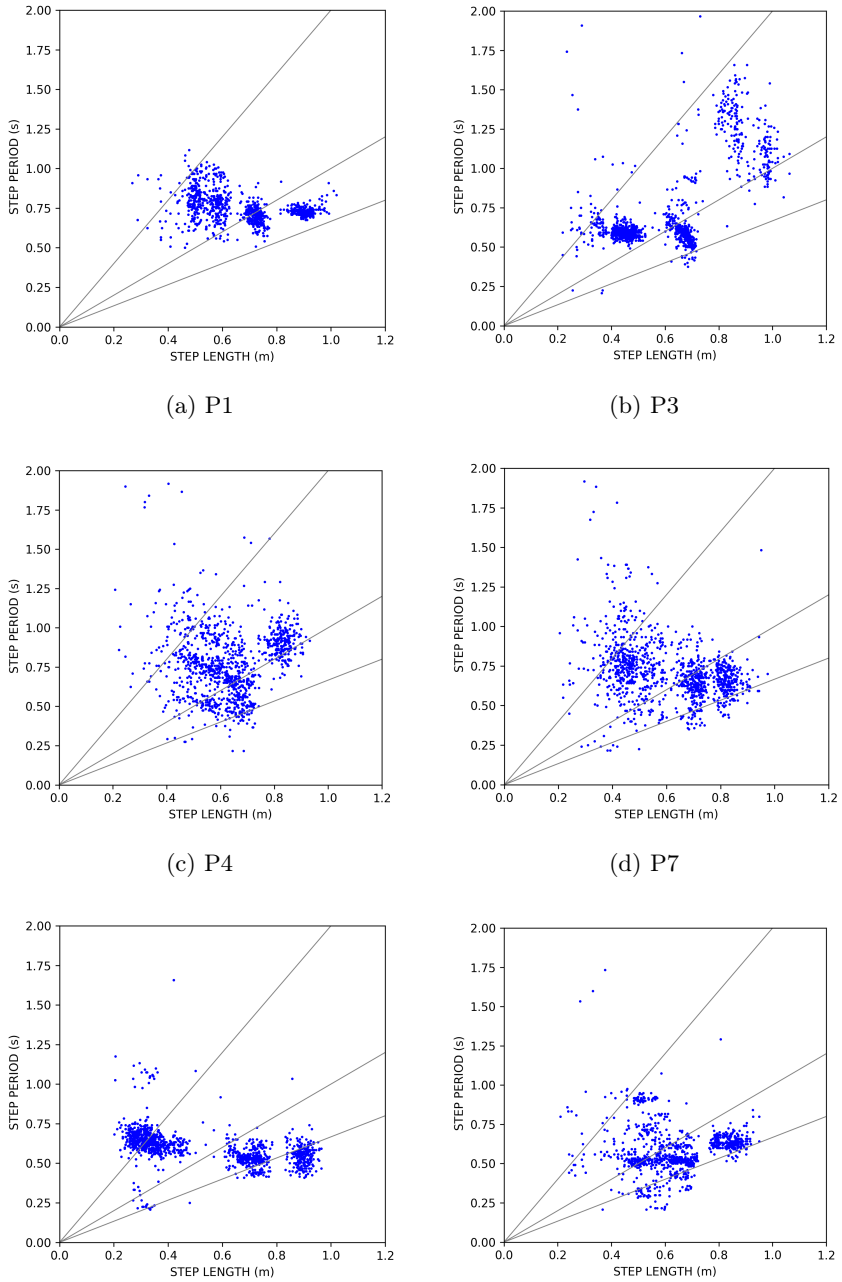


Figure 4.6: Distribution of stride lengths for all participants in our data collection (purple bars), alongside the stride length distribution from the data set described in [110] (green bars).



(a) P1

(b) P3

(c) P4

(d) P7

(e) P9

(f) P12

Figure 4.7: Step period vs. step length for six of our participants in our study. Loci of constant walking speed (0.5 m/s, 1 m/s, and 1.5 m/s) are shown by gray lines.

Figure 4.7 presents scatterplots of step period $\hat{T}_i$ vs. step lengths $\hat{l}_i$ for six of our participants. The figure also shows loci of constant walking speed. These distributions vary among participants. For example, P1 and P9 maintained nearly constant step periods for different step lengths, resulting in significant variations in walking speed. Conversely, P4 and P12 adopted different step periods for different sub-paths with prescribed "shorter than normal" stride lengths, leading to varying walking speeds for the same step length.

4.3 Algorithms

The objective of our system is to estimate either the length l_i or the walking speed v_i during each step, based on inertial data recorded by each smartphone. We use exactly the same architecture for both estimations (l_i and v_i) and assess the results using similar metrics.

Following [110], we implemented a 1-layer LSTM network [46], with 64 hidden units, followed by four fully connected layers with ReLU activation (see Figure 4.8). A recurrent network is the most natural choice for this type of quasi-periodic signal. meter and three gyro measurements. Inspired by [45], we normalize the orientation of these vectors by pre-multiplying them with the inverse of the attitude matrix, provided by the iPhone API. Heel strike times, signifying the start and end of each step, are detected using the step counter LSTM network introduced in the previous chapter. Unlike [110], we maintain the LSTM state without resetting it at the beginning of each step, allowing the network to adapt more effectively to the periodic variations in the input data.

During training, we input segments of a fixed length, consisting of 240 samples, to the LSTM network. Unlike [110], we do not constrain these segments to align with individual strides, which could necessitate zero-padding when dealing with shorter stride periods. Instead, we sample these segments from anywhere in the signal. Specifically, we segment the input signal into intervals of 240 samples with overlap of 120 samples. For each segment, a quadratic loss is defined on the difference between the last output value generated by the network and the ground truth step length $\hat{l}_i$ or walking speed $\hat{v}_i$ associated with the step that contains the end point of the input segment. Our experiments consistently demonstrated that this approach yields superior results compared to using input segments exclusively from individual step periods. The training was performed in Keras, with a batch size of 128 over a total of 500 epochs. To prevent overfitting, training was terminated if if there was no observed reduction in loss over 50 consecutive epochs.

At deployment, the network generates one output sample for each input sample. Figure 4.9 shows examples of the network's output for calculating step length (a) and walking speed (b), along with the detected heel strike times. The plots also display the ground truth values. It is evident from the figure that the network's output exhibits significant variations from one sample to another. Rather than picking one value from the output (e.g., at heel strike times), we compute the average value over each step period to produce the quantity of interest (step length or walking speed). . This is depicted by the blue dashed segments in Figure 4.9.

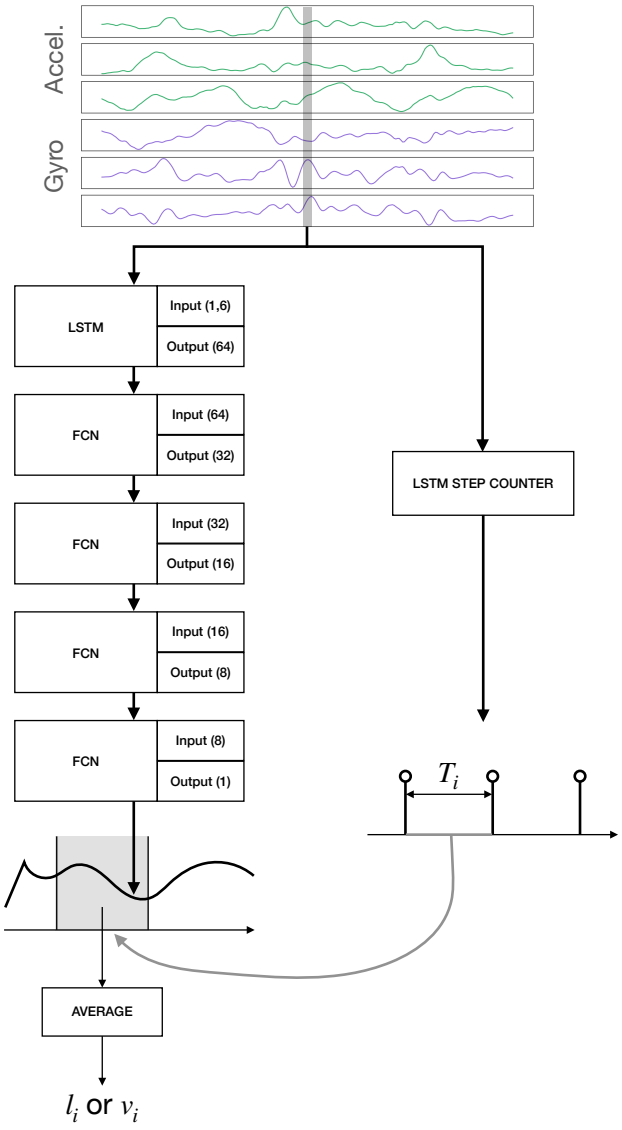
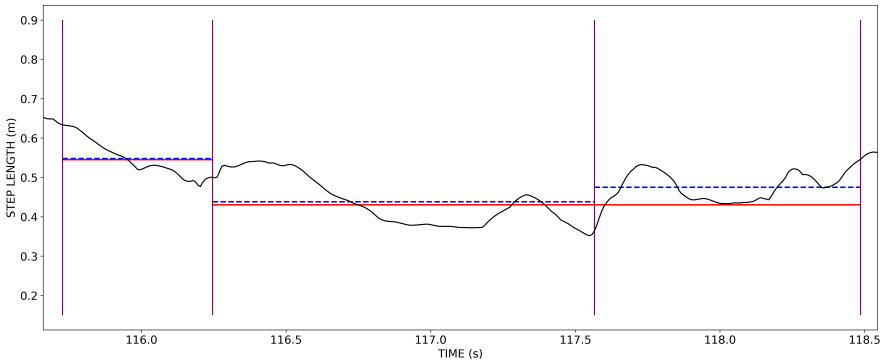
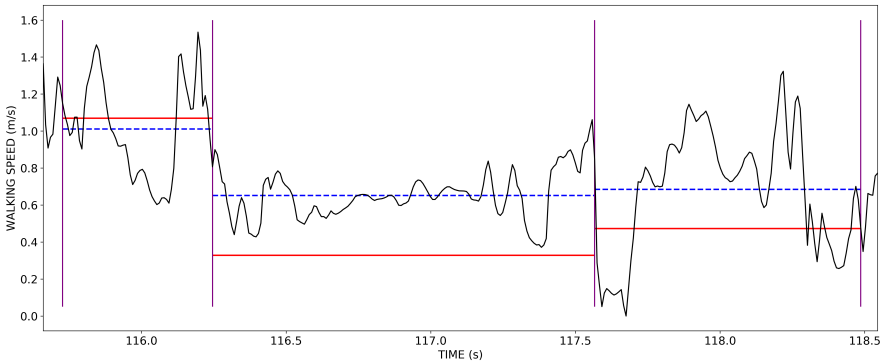


Figure 4.8: The architecture of the network predicting step length l_i or walking speed v_i.



(a)



(b)

Figure 4.9: The network's output predicting either step length (a) or walking speed (b) is shown in black line. Vertical lines indicate detected heel strikes. Red segments represent the ground truth values. Blue dashed segments denote the average output values within a step period. Note that the participant was taking a turn in the path, walking speed significantly reduced in the second and third steps.

4.4 Results

We ran experiments using the leave-one-person-out modality [57]: for each participant being tested, the network was trained with data from the other eleven participants. This approach helps ensure that the network does not overfit to the data from the participant being tested. To assess the impact of phone placement, we considered four scenarios: (1) training and testing with data from the in-hand phone ($H \rightarrow H$); (2) training and testing with data from the in-pocket phone ($P \rightarrow P$); (3) training with data from both phone placements and testing with data from the in-hand phone ($HP \rightarrow H$); (4) training with data from both phone placements and testing with data from the in-pocket phone ($HP \rightarrow P$). For each scenario, we trained two networks: one to estimate step lengths $\{l_i\}$, and another to estimate walking speeds $\{v_i\}$ at each step. The step periods T_i were computed using our LSTM step counter [81] and are identical for the two measurements.

4.4.1 Error Metrics

Remember that $\hat{l}_i$ represents the ground truth length of the i-th step in the test set for a certain participant. We define the following error metrics for the estimated step lengths (where the first and third metrics are from [110]):

- $E_d = \frac{|\sum_{i=1}^{N} l_i - \sum_{i=1}^{N} \hat{l}_i|}{\sum_{i=1}^{N} \hat{l}_i}$
- $E_s = \frac{1}{N} \sum_{i=1}^{N} |l_i - \hat{l}_i|$

- $E_{sr} = \frac{1}{N}\sum_{i=1}^{N}\frac{|l_i - \hat{l}_i|}{\hat{l}_i}$

- $R^2 = 1 - \frac{\text{RMSE}^2}{\sigma^2}$, where $\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(l_i - \hat{l}_i)^2}{N}}$ and σ^2 is the variance of the set of ground truth step lengths $\{\hat{l}_i\}$.

E_d, representing the relative distance error, is particularly applicable to long paths where step-to-step error fluctuations tend to cancel out. E_s denotes the average absolute error at each step, while E_{sr} normalizes errors with the ground truth step length. R^2, the coefficient of determination, is a numeric value that ranges from 0 to 1, reaching 1 only in case of zero error. A negative R^2 indicates that using a constant value, equal to the average step length, would yield a lower RMSE error than the predictions $\{l_i\}$.

Table 4.1 reports the errors measured for the network predicting step lengths l_i. For these and other measurements, each error metric is calculated for each participant, then averaged over all participants. Additionally, standard deviations are reported, computed across participants. The lowest errors are achieved for the $H \rightarrow H$ phone placement configuration, while errors increase for the $P \rightarrow P$ configuration. Training the network with data from both phones ($HP \rightarrow H$ and $HP \rightarrow P$) demonstrates a further decrease in performance. It's important to highlight that the coefficient of determination (R^2) is consistently positive and reaches a value of 0.76 for the $H \rightarrow H$ configuration.

	Step Length			
	E_d	E_s (m)	E_{sr}	R^2
$H \rightarrow H$	0.02 ± 0.02	0.06 ± 0.01	0.10 ± 0.02	0.76 ± 0.12
$P \rightarrow P$	0.05 ± 0.06	0.07 ± 0.03	0.12 ± 0.04	0.61 ± 0.32
$HP \rightarrow H$	0.05 ± 0.03	0.07 ± 0.02	0.12 ± 0.03	0.68 ± 0.19
$HP \rightarrow P$	0.06 ± 0.06	0.08 ± 0.03	0.13 ± 0.04	0.59 ± 0.31

Table 4.1: Error metrics computed for all phone placement configurations for the network predicting step lengths l_i.

For the network predicting walking speed v_i, we present error measures in Tab. 4.2 that are computed based on *equivalent step lengths* $l_i = v_i \cdot T_i$, where T_i is calculated by the step counting network. Essentially, equivalent step lengths are determined by integrating the predicted walking speed over a step period. Additionally, we provide error metrics E_s and R^2 for the walking speed v_i itself when compared to the ground truth $\hat{v}_i = \hat{l}_i / \hat{T}_i$. The results are substantially worse than when predicting step length directly. For instance, the relative distance error E_d for equivalent step length increased by 80% ($HP \rightarrow H$) compared to predicting step length directly. Moreover, the coefficient of determination R^2 for equivalent step length is, in most cases, negative. This suggests that using the mean value would provide a more accurate prediction in terms of mean square error. Paired t-tests revealed that for all the considered metrics, there was a significant difference in error between predicted step length and equivalent step length ($p < 0.02$).

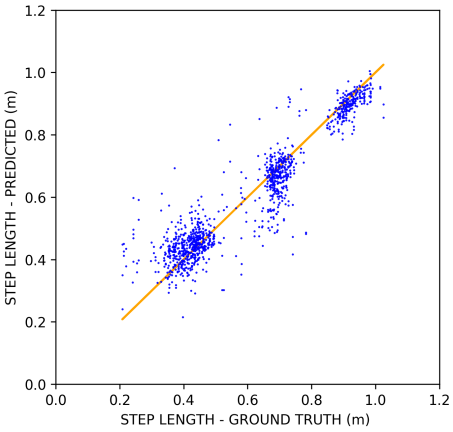
	Step Length From Walking Speed				Walking Speed	
	E_d	E_s (m)	E_{sr}	R^2	E_s (m/s)	R^2
$H \rightarrow H$	0.04 ± 0.02	0.11 ± 0.02	0.20 ± 0.03	0.05 ± 0.39	0.25 ± 0.12	0.24 ± 0.27
$P \rightarrow P$	0.09 ± 0.05	0.13 ± 0.04	0.22 ± 0.08	-0.38 ± 1.22	0.23 ± 0.05	0.22 ± 0.26
$HP \rightarrow H$	0.09 ± 0.04	0.12 ± 0.02	0.22 ± 0.04	-0.14 ± 0.71	0.27 ± 0.12	0.14 ± 0.29
$HP \rightarrow P$	0.08 ± 0.07	0.13 ± 0.04	0.23 ± 0.06	-0.50 ± 1.22	0.26 ± 0.10	0.12 ± 0.25

Table 4.2: Error metrics computed for all phone placement configurations for the network predicting walking speed v_i.

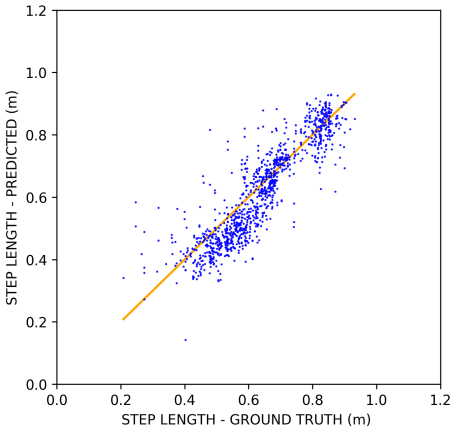
Since equivalent step lengths are determined by integrating the predicted walking speed over the measured step period, computed by our LSTM step counter, it is possible that errors in equivalent step length might be, at least partially, due to inaccuracies in step period computation rather than inaccurate walking speed prediction. To investigate this possibility, we recomputed the equivalent step lengths by integrating the predicted walking speed over the ground-truth step period, as determined by the foot sensors. The corresponding error values were not significantly different from those obtained using step periods from the LSTM step counter.

Figure 4.10 presents scatterplots with step length predictions for different participants and phone placement configurations. The predictive quality is very good in the top two plots (P8: $H \rightarrow H$; P4: $HP \rightarrow H$). However, in the third plot (P9: $P \rightarrow P$), the system exhibited an overestimation of step length by approximately 5 cm for shorter than normal strides and an underestimation by roughly 10 cm for longer than

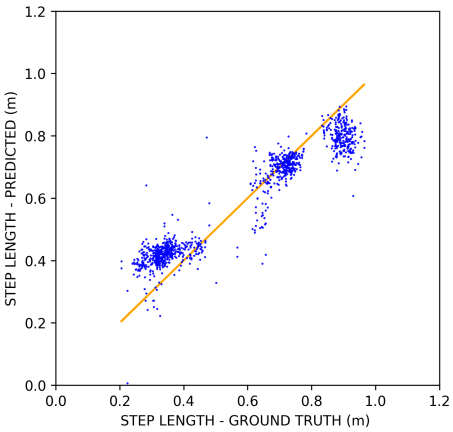
normal strides. The final plot (P1: $HP \rightarrow P$) displayed a consistent negative error that increased with step length.



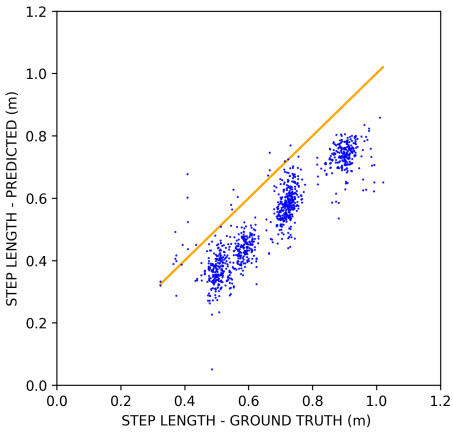
P8: $H \rightarrow H$ ($E_{sr} = 0.09$)



P4: $HP \rightarrow H$ ($E_{sr} = 0.10$)



P9: $P \rightarrow P$ ($E_{sr} = 0.16$)

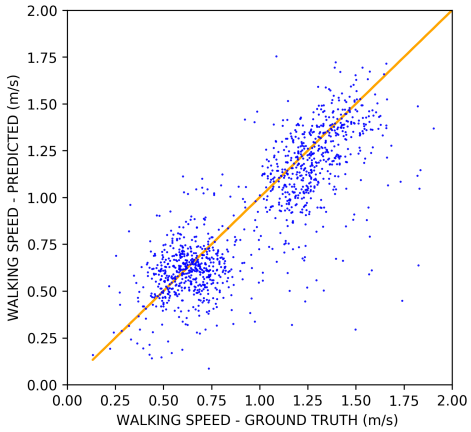


P1: $HP \rightarrow P$ ($E_{sr} = 0.22$)

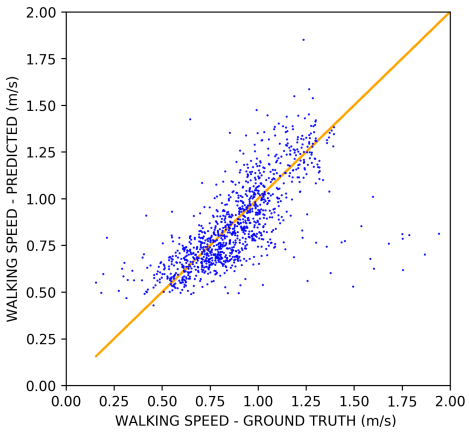
Figure 4.10: Examples of step length prediction, plotted against their ground truth values.

Figures 4.11 and 4.12 present the results of walking speed estimation (v_i) and

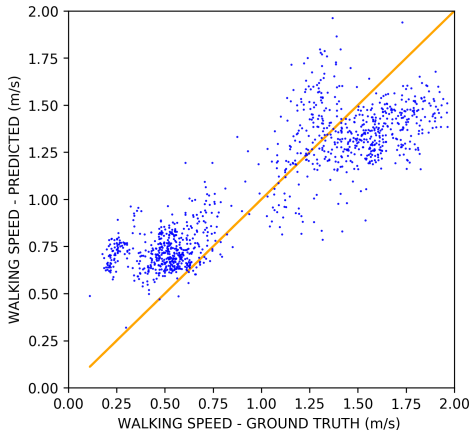
equivalent step length prediction using the same input data. The system performance clearly deteriorates with respect to the prior case, as confirmed by the higher values of E_{sr} as recorded in the Figures 4.10 and 4.12.



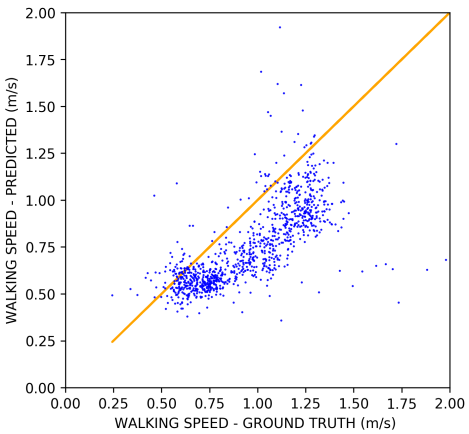
P8: $H \rightarrow H$



P4: $HP \rightarrow H$

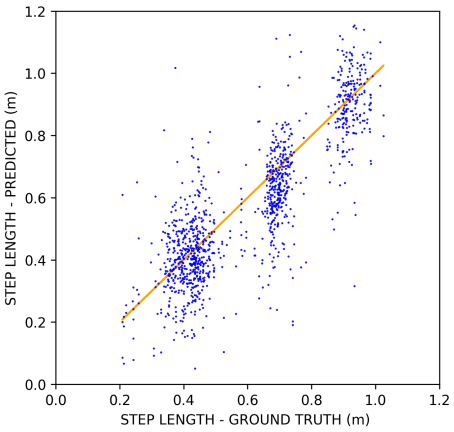


P9: $P \rightarrow P$

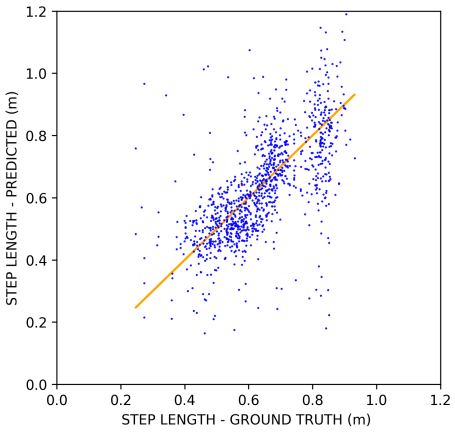


P1: $HP \rightarrow P$

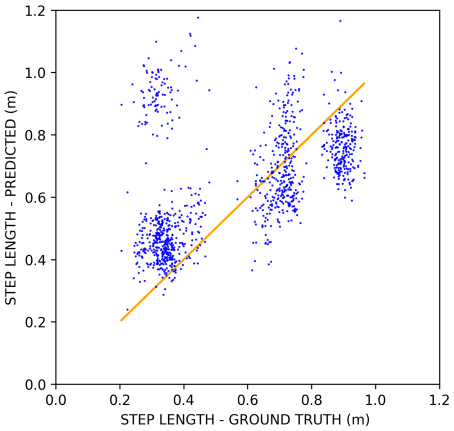
Figure 4.11: Examples of walking speed predictions, plotted against their ground truth values.



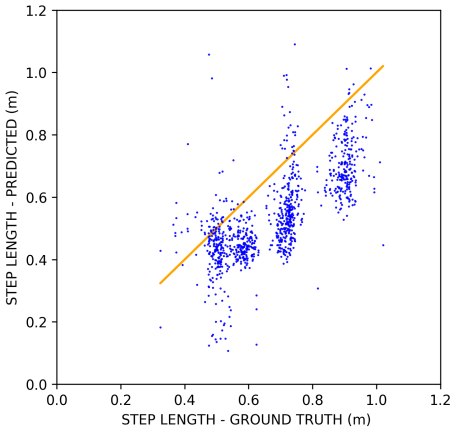
P8: $H \rightarrow H$ ($E_{sr} = 0.17$)



P4: $HP \rightarrow H$ ($E_{sr} = 0.15$)



P9: $P \rightarrow P$ ($E_{sr} = 0.39$)



P1: $HP \rightarrow P$ ($E_{sr} = 0.23$)

Figure 4.12: Examples of step length prediction derived from walking speed predictions, plotted against their ground truth values.

4.4.2 Comparison with RoNIN

In order to assess the quality of our LSTM-based algorithm, we conducted a comparative analysis with RoNIN, a state-of-the-art pedestrian dead-reckoning system, using the same iPhone sensor data. RoNIN is designed to compute the user's velocity relative to a fixed reference frame. To make a fair comparison, we integrated the velocity data from RoNIN over each step period, which was measured by our LSTM step counting model [81]. The resulting vector length was then extracted and compared with the ground truth length from the foot sensor. We employed the open-source implementation provided by the authors of RoNIN (https://github.com/Sachini/ronin) and adopted the RoNIN resnet18 architecture. To ensure consistency, we up-sampled the data from its original acquisition rate of 120 Hz to match the 200 Hz sampling rate used for the RoNIN design. We obtained results with both RoNIN and Scaled-RoNIN, which is a customized version of RoNIN as explained in Section 3.4.3. We determined a scale factor α by minimizing the mean squared error between the step length from RoNIN, multiplied by α, and the ground-truth step length ($\alpha = 1.15$ and 1.24 for in-hand and in-pocket phone data, respectively).

Tables 4.3 and 4.4 present the error measures for RoNIN and Scaled-RoNIN, respectively, using data from in-hand (H) and in-pocket (P) phones. The results show a performance increase when using Scaled-RoNIN. Notably, the predicted step length from our system (Table 4.1) exhibited substantially lower errors compared to the equivalent step length from both RoNIN (Table 4.3) and Scaled-RoNIN (Table 4.4). Additionally,

our LSTM network's predictions for walking speed demonstrated superior performance compared to RoNIN, achieving better results on in-hand phone data and comparable results on in-pocket phone data compared to Scaled-RoNIN.

	Step Length From Walking Speed				Walking Speed	
	E_d	E_s (m)	E_{sr}	R^2	E_s (m/s)	R^2
H	0.19 ± 0.07	0.14 ± 0.04	0.25 ± 0.09	-0.26 ± 0.42	0.30 ± 0.15	0.06 ± 0.29
P	0.23 ± 0.07	0.17 ± 0.04	0.27 ± 0.05	-0.79 ± 1.08	0.31 ± 0.10	-0.12 ± 0.31

Table 4.3: Error metrics computed for two phone placement configurations using RoNIN.

	Equivalent Step Length From Walking Speed				Walking Speed	
	E_d	E_s (m)	E_{sr}	R^2	E_s (m/s)	R^2
H	0.08 ± 0.06	0.12 ± 0.03	0.23 ± 0.08	-0.05 ± 0.37	0.27 ± 0.14	0.16 ± 0.31
P	0.08 ± 0.06	0.11 ± 0.05	0.19 ± 0.07	-0.27 ± 1.06	0.23 ± 0.11	0.22 ± 0.31

Table 4.4: Error metrics computed for two phone placement configurations using Scaled-RoNIN.

4.4.3 Model Performance on a Different Data Set

As explained in Section 4.3, we employed a different training methodology and made adjustments to the model architecture compared to the approach presented in [110]. To assess our model's performance, we conducted a 10-fold cross-validation on the same data set used in [110]. The results are reported in Table 4.5. Our system demonstrated improved results when compared to [110], particularly in terms of the mean E_{sr} error

rate. Additionally, we achieved a substantial reduction in the mean E_d error compared to both models considered in [110] (LSTM and LSTM-DAE). Specifically, our system reduced E_d from 0.05 to 0.01 when compared to their LSTM model and from 0.04 to 0.01 when compared to LSTM-DAE.

Models	E_d	E_{sr}
LSTM (ours)	0.01	3.02%
LSTM [110]	0.051	3.75%
LSTM-DAE [110]	0.043	3.16%

Table 4.5: Comparison of error metrics between our step length prediction algorithm and the algorithms described in [110] using the data set from [110].

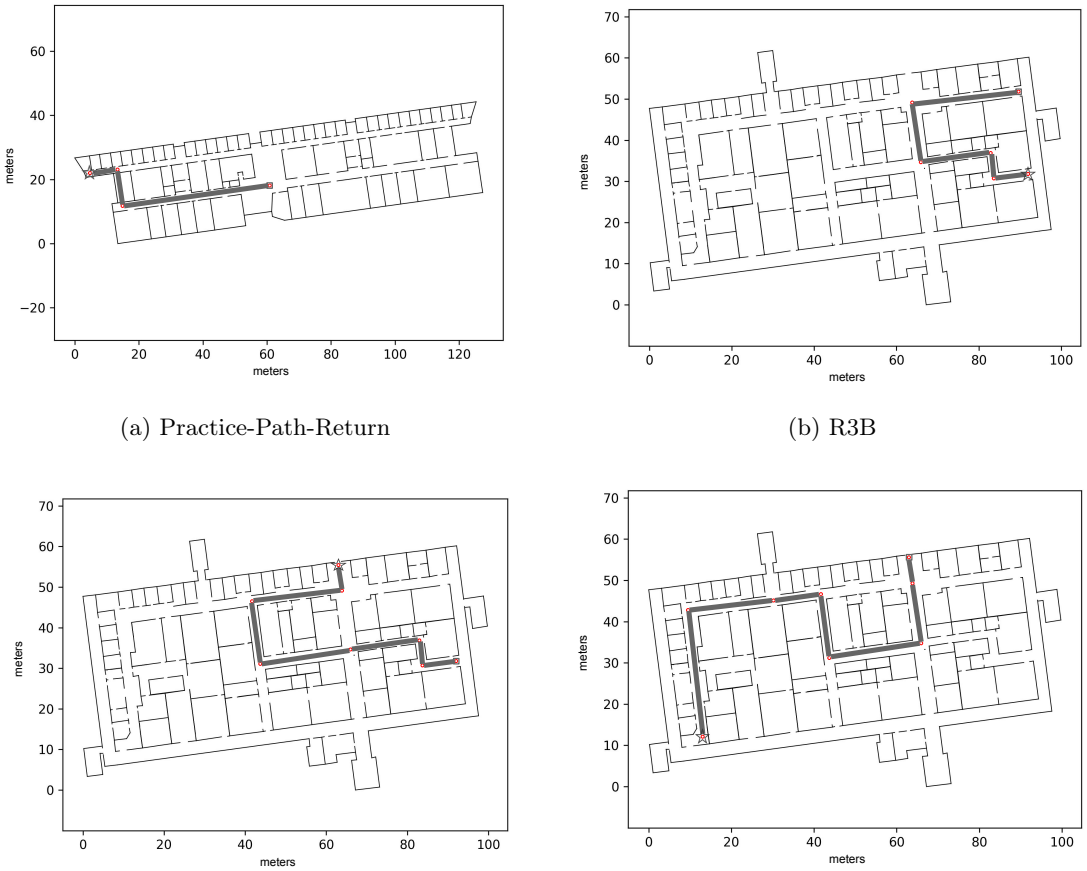
4.5 Step Length Estimation for Visually Impaired Subjects

4.5.1 Data Collection

At the time when we were collecting data for this project, we were unable to invite blind participants to attend our study due to COVID-19 restrictions. However, in a subsequent user study to evaluate our iOS applications, which will be discussed in Chapter 5, we attached the same foot sensors (Xsens DOT), securing each to the participants' shoes using an elastic band. Additionally, we instructed participants to keep an iPhone 12 in their pants pocket while traversing the return routes to collect phone inertial data. This approach allowed us to synchronize the inertial data from both the phone and the foot sensors. Notably, blind users typically prefer not to hold phones

in their hands, as they already have one hand occupied holding a cane or a dog's leash. Consequently, unlike our data collection from sighted users, we collected data with the phone exclusively placed in the pants pocket, without using two phones – one in hand and the other in the pants pocket.

We will delve into the details of the user study experiment in the following chapter. However, to provide context, I explain the routes taken by each participant. We designed a total of four routes: one is referred to as the 'Practice-Path' in the Enginnering 2 (E2) building, and the other three routes are located in the Baskin Engineering Building (R1W, R2W, and R3W). Each participant initially traversed the practice route, after which we attached foot sensors and instructed them to keep a phone in their pocket to collect inertial data as they retraced the same route in reverse (Practice-Path-B). The same procedure applied to the routes in the Baskin Engineering (BE) Building. Participants first navigated R1W, followed by R2W, and finally R3W. Subsequently, they retraced these routes in the reverse order, which we refer to as R3B, R2B, and R1B (see Figure 4.13). A total of seven participants took part in this study, including 5 cane users and 2 guide dog users. We did not record phone inertial data for the first user on the three routes in the BE building. To compensate for this loss of data, we recorded the path the user took from the BE building to the office in the E2 building. It's also important to note that, due to the study's main objectives, some participants could not complete entire routes in certain trials (Figure 4.14 (b)), and others may have made incorrect turns but subsequently corrected their path to reach their destination (Figure 4.14 (c)).



(a) Practice-Path-Return

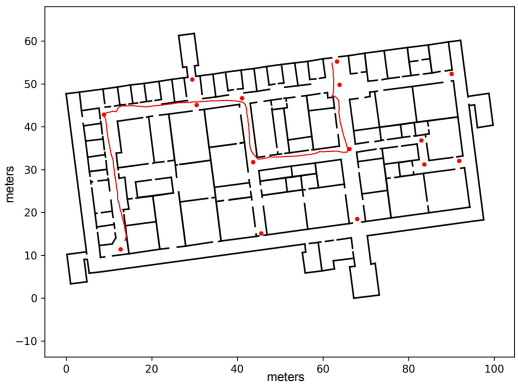
(b) R3B

(c) R2B

(d) R1B

Figure 4.13: ((a)–(d))The four routes that blind participants traversed in the study. The start and end positions are marked with a square and a star, respectively.

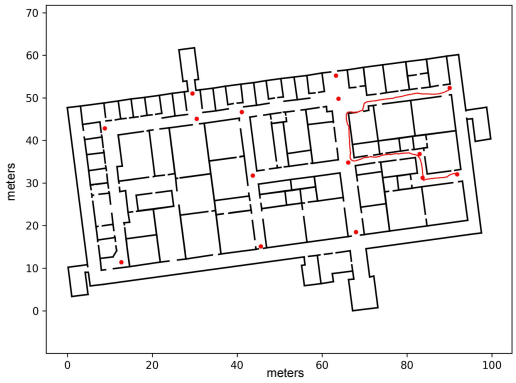
In order to establish the ground truth for the users' step lengths, we employed the EKF algorithm, as previously explained in Section 4.2.1.5. Please note that despite the ZUPT and HDR corrections, this algorithm still experiences drift due to the well-known issue of sensor noise (Figure 4.14 (a)). More samples of the paths reconstructed using our EKF algorithm are shown in Figure 4.14.



(a) P3, R1B

(b) P6, R3B

(c) P2, R2B



(d) P3, R3B

Figure 4.14: ((a)–(d)) Sample reconstructed trajectories (in red) of different routes traversed by various participants, obtained by our EKF algorithm, superimposed on the floor plan of the BE building. (a) The reconstructed trajectory of route R1B traversed by P3 shows signs of drift. (b) The reconstructed trajectory of route R3B traversed by P3, but he was unable to reach his destination. (c) The reconstructed trajectory of route R2B traversed by P2, including a wrong turn but eventual correction. (d) A successful traversal of route R3B by P3.

Figure 4.15 illustrates the distribution of stride lengths measured for all blind participants during our user study (displayed in orange) (mean: μ =0.86m, standard deviation: σ =0.23m). The same figure also presents the distribution of stride lengths in the data set collected from sighted participants (shown in purple) (mean: μ =1.2m, standard deviation: σ =0.38m) using the same foot-mounted sensors, as described in Section 4.2. Notably, the mean stride length for blind participants is shorter than that of sighted individuals.

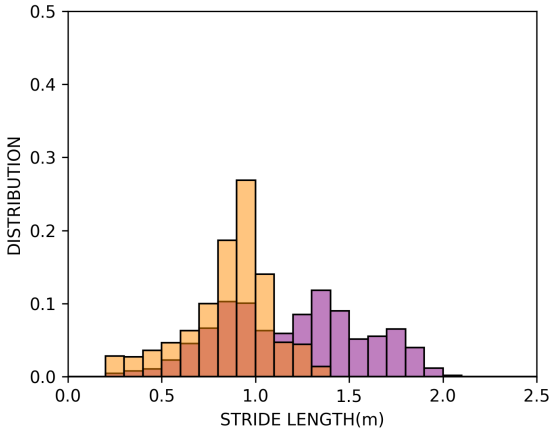
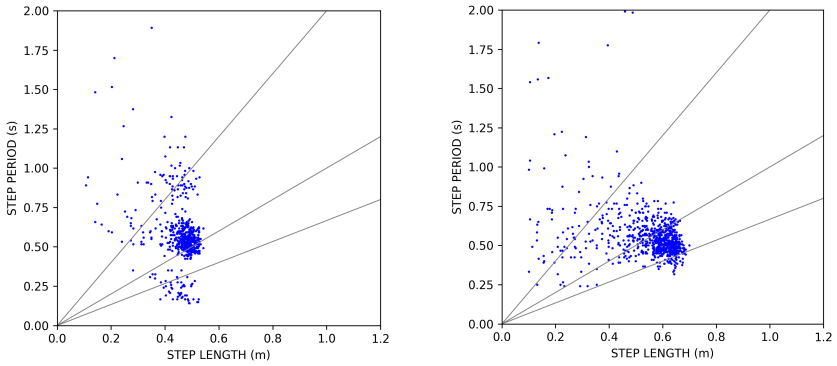


Figure 4.15: Distribution of stride lengths for all blind participants in our study (orange bars) is presented alongside the stride length distribution for the data set of sighted participants (purple bars).

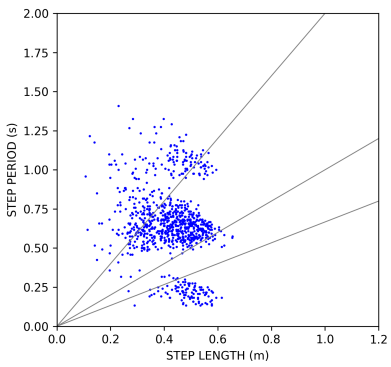
Our objective is to predict step lengths using smartphone inertial data carried in users' pants pockets. As previously mentioned, smartphone data is periodically recorded within each step, with one foot in the stance phase and the other in the swing phase. Similar to our data set with sighted users, we establish ground truth step lengths by averaging two overlapping stride lengths (one for each foot) and dividing the result by 2.

Figure 4.16 displays scatterplots of step period $\hat{T}_i$ versus step lengths $\hat{l}_i$ for our participants who used a guide dog as their navigation aid. Whereas Figure 4.17 displays scatterplots for our participants who used a white cane as their navigation aid. It's important to observe that these distributions vary among participants. For instance, P1 and P7, both guide dog users, show distinct stride patterns. P1 typically takes shorter steps, while P7 walks at a faster pace with longer strides. Among cane users, P5 tends to have shorter steps on average, with a wide range of step lengths, while P3 takes longer strides on average.

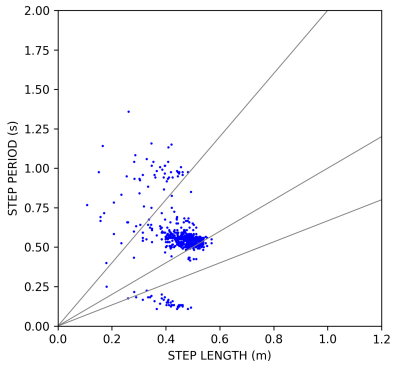


(a) P1 (b) P7

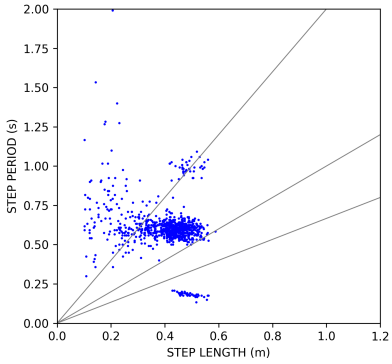
Figure 4.16: Step period vs. step length for our participants who use a guide dog as navigation aid in our study. Loci of constant walking speed (0.5 m/s, 1 m/s, and 1.5 m/s) are shown by gray lines.



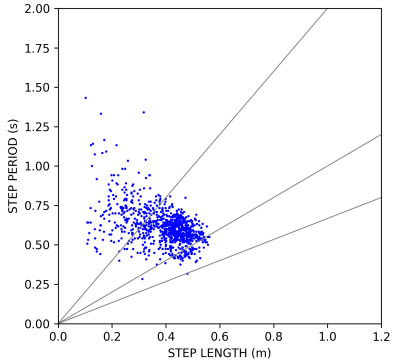
(a) P2



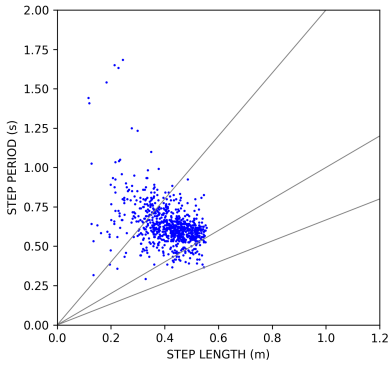
(b) P3



(c) P4



(d) P5



(e) P6

Figure 4.17: Step period vs. step length for our participants who use a cane as navigation aid in our study. Loci of constant walking speed (0.5 m/s, 1 m/s, and 1.5 m/s) are shown by gray lines.

4.5.2 Results

We utilized the LSTM model explained in Section 4.3 for predicting the step length of blind individuals using data from the smartphone carried in their pants pockets.

4.5.2.1 Training/Test Modalities

The gait patterns exhibit notable differences between individuals using white canes and those relying on guide dogs [114, 47]. These distinctions were clearly observed in our experiments, as illustrated in the results detailed in the previous chapter. Moreover, they are visually evident in the scatter plots depicting step lengths for various users (refer to Figures 4.7, 4.16, and 4.17). To investigate the impact of these differences, and following the training and testing modalities we utilized in Chapter 3, we categorized the results based on the communities of long cane users (indicated by the modifier:LC) and guide dog users (:GD). We employed the following training and testing schemes, ensuring that the system used to evaluate a particular walker had never been trained on that walker's data.

- Train on Sighted (TS): The model is trained using data from the smartphones carried in the pockets of the twelve sighted walkers in our collected data set. Subsequently, the system is tested on two communities of blind users who participated in our user study (TS:LC and TS:DG)

- Train in the same Community (TC): In this scenario, the system was tested with long cane users and guide dog users, and it was trained using data from walkers

within the same community (TC:LC, TC:GD). We employed the Leave-One-Person-Out (LOPO) policy to perform this training and testing approach [37, 57]. With LOPO, each participant was tested with a system trained on data from all other participants within the same community. This training approach allowed us to explore how the walking characteristics may differ between communities of users. It's worth noting that only two walkers in our user study used a guide dog, so each training set in the TC:GD modality contained data from a single walker only.

- Train on Blind (TB): In this case, all data collected from blind participants in our user study was used for training, following the Leave-One-Person-Out (LOPO) policy (TB:LC, TB:DG). For instance, a long cane user is tested with a system trained on data from all guide dog users and all other long cane users.

- Train on All (TA): In this case, all the data collected from the twelve sighted walkers in the data set, as well as the seven blind walkers who participated in the user study, were used for training, following the Leave-One-Person-Out (LOPO) policy (TA:LC, TA:DG). For instance, a long cane user is tested with a system trained on data from all sighted participants, all guide dog users, and all other long cane users.

We have utilized the same error metrics as explained in Section 4.4.1 and are reporting the results separately for the white cane users' community in Table 4.6 for all Training/Test modalities and in Table 4.7 for guide dog users' community for all modalities. Each error metric is computed for each participant, then averaged over the

participants. We also report standard deviations computed across participants.

	E_d	E_s (m)	E_{sr}	R^2
$TS : LC$	0.14 ± 0.10	0.09 ± 0.03	0.29 ± 0.11	-0.95 ± 0.96
$TC : LC$	0.02 ± 0.01	0.05 ± 0.01	0.14 ± 0.03	0.48 ± 0.08
$TB : LC$	0.03 ± 0.02	0.05 ± 0.01	0.14 ± 0.03	0.43 ± 0.12
$TA : LC$	0.02 ± 0.01	0.05 ± 0.01	0.15 ± 0.03	0.44 ± 0.03

Table 4.6: Error metrics computed for all Training/Test modalities for cane users for the network predicting step lengths l_i.

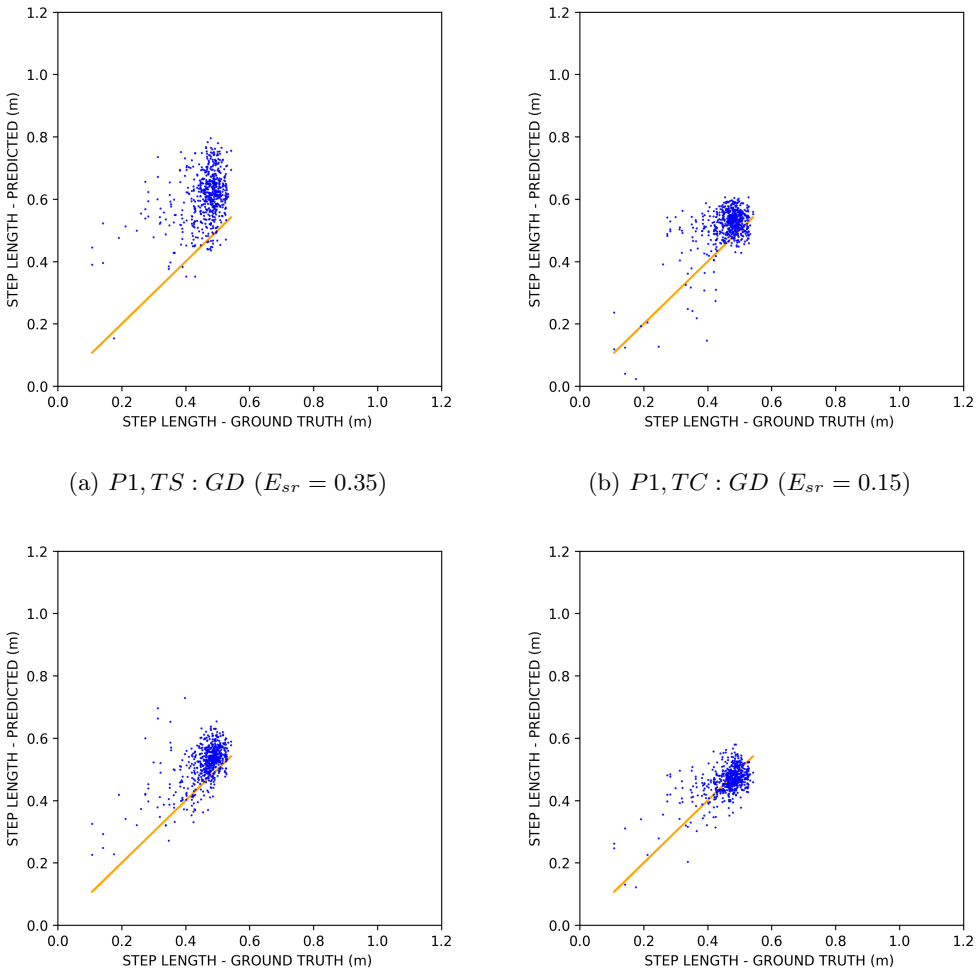
	E_d	E_s (m)	E_{sr}	R^2
$TS : GD$	0.19 ± 0.18	0.14 ± 0.01	0.33 ± 0.03	-3.79 ± 4.64
$TC : GD$	0.18 ± 0.10	0.11 ± 0.07	0.23 ± 0.11	-0.75 ± 0.03
$TB : GD$	0.07 ± 0.10	0.07 ± 0.04	0.15 ± 0.08	0.25 ± 0.07
$TA : GD$	0.07 ± 0.08	0.07 ± 0.01	0.17 ± 0.02	-0.22 ± 0.94

Table 4.7: Error metrics computed for all Training/Test modalities for guide dog users for the network predicting step lengths l_i.

The results presented in Tables 4.6 and 4.7 clearly illustrate the impact of the walker community used for system training on the performance of the step length prediction model. For instance, when the model was trained on sighted users, it yielded the poorest results among all modalities when tested on GD users (TS:GD) (see Figure 4.18 (a)), with negative R^2 coefficient. Additionally, due to the limited number of guide dog users in our user study (only one user in each training set round, and testing

on the other user), and considering the vastly different gait patterns of these two users, poor results were still obtained when the model was trained on GD users and tested on the remaining one ($R^2 = -0.75$). Training on all users and testing on guide dog users in the LOPO manner still results in a negative $R^2 = -0.22$, indicating that the walking patterns of sighted walkers and GD users are not similar, as observed in the first row of the results table in Table 4.7. However, excluding the sighted walkers from the training set improves the results and yields the best performance for GD users with an $R^2 = 0.25$ (see Figure 4.18 (b–c)).

Similar to the results for GD users, the model trained on sighted users and tested on cane users exhibits significantly higher error values, indicating poor prediction performance. The negative R^2 value of -0.95 suggests that the predictions do not fit well with the actual step lengths of cane users. However, in contrast to GD users, we had 5 subjects who used a long cane as their navigation aid in our user study, and they exhibited similar gait patterns. When the model was trained on cane users and tested on the remaining one, it showed strong performance with a low E_d value as low as 0.02, and an R^2 value reaching 0.48. For the models trained on a combination of white cane users and guide dog users, or trained on all data and tested on white cane users, the error values are slightly higher compared to the WC:WC scheme, indicating slightly less accurate predictions but still fairly good (see Figure 4.19). In these cases, the R^2 values are 0.43 and 0.44, respectively.



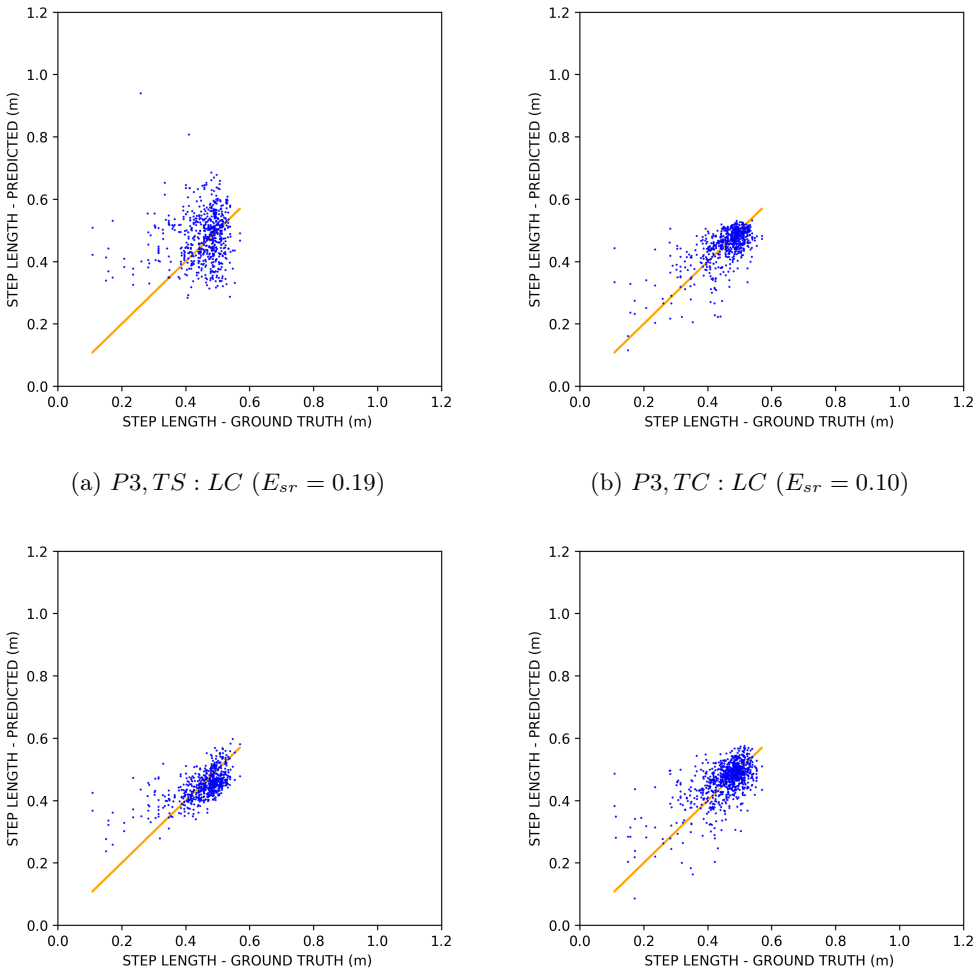
(a) $P1, TS : GD$ ($E_{sr} = 0.35$)

(b) $P1, TC : GD$ ($E_{sr} = 0.15$)

(c) $P1, TA : GD$ ($E_{sr} = 0.15$)

(d) $P1, TB : GD$ ($E_{sr} = 0.09$)

Figure 4.18: Examples of step length prediction for $P1$ for different Training/Test modalities, plotted against their ground truth values.



(a) $P3, TS : LC$ ($E_{sr} = 0.19$)

(b) $P3, TC : LC$ ($E_{sr} = 0.10$)

(c) $P3, TA : LC$ ($E_{sr} = 0.11$)

(d) $P3, TB : LC$ ($E_{sr} = 0.11$)

Figure 4.19: Examples of step length prediction for $P3$ for different Training/Test modalities, plotted against their ground truth values.

4.6 Conclusions

PDR systems for the reconstruction of odometry from inertial data from a smartphone measure the distance traveled during a path through the detection of

individual steps and estimation of their lengths. Assuming reliable step detection, one can choose to predict either the actual step length or the average walking speed for each step. Existing literature addresses these two tasks differently: step length prediction typically relies on data from individual steps, while velocity prediction (e.g. [45]) is often calculated at a higher frequency and then integrated over time to derive position information. We employed the same computational architecture to estimate both step length and average walking speed for each step. To do so, We curated a carefully annotated data set, including participants walking with diverse stride lengths. We adopted a Leave-One-Person-Out approach for training and testing the system. Our results strongly support the conclusion that predicting step length is more reliable than predicting average walking speed.

It can be argued that the dynamics of the body during walking, as measured by the smartphone's inertial sensors, might exhibit a stronger dependence on stride length than on the pacing rate. The distribution of step length and step period could also play a role in the prediction results. Some insight can be gained by analyzing the *coefficient of variation (CV)*, which is the standard deviation divided by the mean, for the considered quantities (step length, step period, and their ratio, representing average walking speed in each step). These CV values were computed for each participant and each prescribed stride length, and then averaged across participants, considering only ground-truth data for this analysis. It was observed that the CV values for step lengths (0.12, 0.09, and 0.08 for short, natural, and long strides, respectively) are considerably lower than those obtained for step periods (0.35, 0.42, 0.29) and, consequently, for walking speed (0.46,

0.54, 0.25). Note that the CV values for step length, while slightly higher, may be attributed to the variations in experimental settings compared to those reported in [91]. Arguably, a recurrent network tasked with tracking a quantity that is locally "stable" (step length) may face a less challenging task compared to predicting a quantity that exhibits greater variation from step to step (walking speed).

Our results (Table 4.1) indicated that improved prediction is achieved when training and testing with a phone placed in the same location on the user's body. This insight could be valuable if a system to detect the phone's placement, such as in [73], is implemented. It allows for the selection of the most appropriate step length predictor based on the phone's current placement. Our analysis focused solely on *length*, not the direction, of steps taken while walking. Existing learning-based approaches like those in [45, 66, 14] can be applied to robustly estimate heading direction.

We also assessed the architecture's performance for predicting step length with visually impaired participants in our user study. Our analysis reinforces the findings presented in the previous chapter, highlighting the critical importance of selecting from the community of walkers for training the algorithm's parameters. The model trained on in-pocket phone data from sighted walkers exhibited lower performance when tested on cane and guide dog users. On the other hand, when training data included these communities, there was a significant improvement in results. Although they didn't reach the same level of performance as TC:S (train on sighted, test on sighted), they achieved comparably good results, despite the smaller number of participants (7 compared to 12).

Chapter 5

WayFinding

5.1 Introduction

Independent navigation is a fundamental aspect of daily life for most individuals, allowing them to move freely, explore new environments, and accomplish various tasks without external assistance. Navigation in unfamiliar surroundings can be especially challenging. Sighted individuals depend on landmarks to help them in wayfinding by providing essential spatial location and task-related information. While they may have a clear destination in mind, the specific spatial coordinates and path to reach it are unknown. Sighted individuals use visible landmarks as guiding points to help them navigate through their environment and reach their intended destination. However, this can be more challenging for people with visual impairments (PVI) due to their lack of confidence and knowledge about the environment [114].

While many visually impaired individuals can independently navigate familiar

routes by forming detailed cognitive maps through exploration and receiving training from Orientation and Mobility (O&M) professionals, they face significant challenges when it comes to new or infrequently visited places where prior exploration is impossible. In such situations, independent travel becomes quite difficult for them. The physical cues that indicate possible paths or points of interest in the surroundings are usually visual, like distant signs or landmarks. Hence, they are inaccessible to people with visual impairment. Moreover, people with visual impairments may lack awareness of their position within a building's floor plan and their relationship to essential features such as stairs, doors, elevators, and obstacles. Consequently, in such scenarios, seeking assistance from a passerby to find an accessible route becomes necessary, but there is no assurance of someone being available to assist at all times [3, 77].

Different assistive navigation technologies are proposed to enhance users' independence by providing guidance to their destination and alerting them about nearby Points-Of-Interests (POIs). Despite their usefulness in supporting navigation for individuals with visual impairments, these systems encounter several limitations and challenges. For instance, GPS-based approaches have been leveraged to provide outdoor navigation. Still, due to their inaccuracy in indoor places and outdoor areas with high construction density [107], there is a need for assistive technology that aids individuals with visual impairments in indoor navigation. Currently, there are no commercially available systems suitable for extensive use. However, researchers have investigated different strategies to facilitate indoor localization and navigation for PVI. These strategies involve using ultrasound, infrared, RFID, and ultra-wide band (UWB) sensors. While these

methods achieve acceptable accuracy, they typically require users to carry a dedicated device. [22, 1, 48, 27, 85].

Researchers are dedicated to creating assistive technologies that free people with visual impairments from the necessity of carrying additional devices, mainly because they often rely on white canes or guide dogs while walking. Smartphone-based approaches have the added benefit of offering localization and navigation capabilities without burdening users with extra devices. These methods can provide guidance using a variety of sensors, such as Bluetooth Low Energy (BLE) beacons, Wi-Fi, inertial measurement units (IMUs), video cameras, or a fusion of these sensors [29, 20, 34, 50, 67, 70, 71, 86].

The availability of accurate and detailed building maps plays a crucial role in ensuring the effectiveness and reliability of such indoor wayfinding systems. These maps provide essential information about the configuration of rooms, hallways, staircases, elevators, and other structural elements necessary for generating accurate navigation routes. Moreover, building maps also contain data about critical points of interest (POIs), such as entrances, exits, restrooms, emergency exits, offices, meeting rooms, and amenities. Access to these POIs is vital as it guides users to specific destinations, enhancing their overall indoor navigation experience. Additionally, indoor wayfinding systems require information about walkable paths, potential obstacles, and potential bottlenecks within the building for pathfinding and routing. Building maps facilitates the generation of optimized routes that avoid obstacles and provide the shortest or most accessible paths for users. Furthermore, the availability of building maps is essential for integrating with various positioning technologies and sensors, such as Wi-Fi, Bluetooth beacons, RFID,

and cameras. Combining map data with sensor inputs can enhance localization accuracy and provide more reliable navigation guidance.

Indoor maps' availability is undoubtedly on the rise, thanks to commercial efforts like Google Indoor Maps. However, it would be impractical to expect accurate maps of all public buildings to be universally accessible in a standardized format. In cases where building maps are unavailable, assisted return strategies can be utilized to guide individuals effectively. Assisted return refers to offering support to a visually impaired user attempting to retrace their steps and find their way back to the starting point after walking along a particular path when human assistance is unavailable. An example application for assisted return could be in a hospital setting. A visually impaired patient receives assistance from hospital staff to navigate from their room to a specific department for a medical appointment. After the appointment, the patient might need to independently return to their room or designated waiting area. In this scenario, the assisted return application would offer support to the visually impaired individual as they make their way back on their own[29, 105].

In the following sections, I will begin with reviewing currently available smartphone-based applications for indoor wayfinding within the related work section. Following that, I will discuss the development of the two distinct iOS applications, which were a collaborative effort with my colleagues from the computer vision lab. These applications have been specifically crafted to offer assistance in both wayfinding and backtracking for blind travelers when they move around indoors. Wayfinding encompasses the process of determining and following a route through a building's hallways to reach a destination.

This process relies on the app having access to the building's floor plan. On the other hand, backtracking involves retracing one's steps without relying on any previous map information. Our apps solely use smartphones' inertial and magnetic sensors, eliminating the need for infrastructure modifications, such as the installation and support of BLE beacons. In contrast to systems that utilize the phone's camera, our apps allow users to conveniently keep their phones in their pockets while interacting with the apps via a smartwatch. Routing directions are delivered audibly. As my primary contribution lies in the wayfinding app, I will provide a more detailed discussion of it within my thesis. Both applications underwent successful testing in a user study that involved seven blind participants navigating a campus building, as explained in the experiment section.

5.2 Related Work

5.2.1 Indoor WayFinding Apps with Maps

5.2.1.1 NavCog (Navigational Cognitive assistant)

NavCog [3] is an advanced smartphone-based turn-by-turn navigation system created collaboratively by researchers from Carnegie Mellon University and IBM research. The system offers real-time navigation assistance across extensive areas, specifically designed to aid individuals with visual impairments while navigating unfamiliar indoor environments. It incorporates a precise localization algorithm that strikes a balance between accuracy and deployment workload, an interactive feature based on customizable voice and non-vocal sound instructions, and tools to expedite the deployment process.

Additionally, the app provides users with information about nearby POIs and accessibility concerns, such as the presence of stairs ahead.

The core of the NavCog system lies in the interplay between three essential components: the Map Server, Beacon Localization, and the NavCog App (see Figure 5.1).

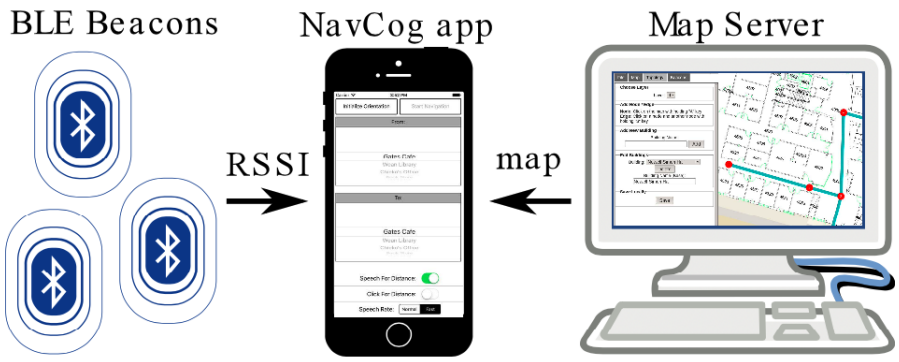


Figure 5.1: The components of the NavCog system: BLE beacons, Map Server, and the NavCog app [3]

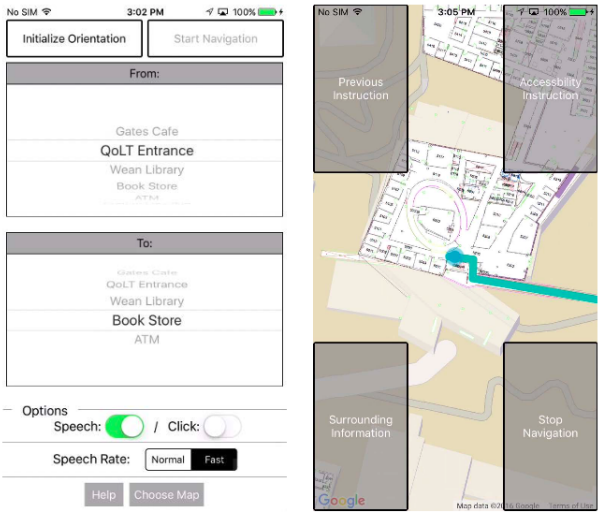
- Map Server: The map-building process for the NavCog app involves several steps. First, a floor plan of the environment is uploaded to a map server. Using a web-based map editor, users mark beacons, walkable areas, decision points, and POIs on the map. The map can be easily updated as the mapped areas expand. Once the map is complete, it is uploaded to a remote server and then retrieved by the NavCog app on users' smartphones. This approach enables offline localization and navigation since the position computation is entirely performed on the mobile device, eliminating the need for a network connection. The map is structured as a graph, with nodes representing significant locations in the environment. Additionally,

the map server handles the management of Bluetooth beacons installed in the environment, which play a crucial role in the localization process.

- Beacon Localization: Beacons provide several advantages, such as more precise localization compared to GPS or WiFi-based methods, easy installation and maintenance, and increasing popularity. To localize the user, NavCog creates a model linking BLE beacon signals to positions in the environment. BLE beacon signals are sampled at known positions during the model training stage using a fingerprint-based method. By employing a K-nearest neighbor (KNN) algorithm with a K-d tree data structure, NavCog efficiently identifies the nearest BLE beacons based on RSSI readings from the user's smartphone. This enables the system to accurately estimate the user's position, ensuring real-time localization for visually impaired individuals during indoor navigation.

- NavCog App: The NavCog App comprises a planning interface and a navigation interface, designed with a user-friendly layout for individuals with visual impairments (see Figure5.2). The planning interface allows users to select destinations and plan routes, change vocal message speed, and the preferred way to receive instructions. The navigation interface aids users during navigation. The design of this interface aims to facilitate simplified interactions between visually impaired users and the touch screen. Positioned at the corners of the touch screen are four strategically located buttons, each with a specific purpose. The bottom right button concludes navigation and returns the user to the planning interface, while the top left button

replays the last navigation message, assisting users who might have missed it due to distractions. The top right button provides accessibility-related guidance based on the user's location, and the bottom left button allows users to request additional information about their surroundings. The interface also includes a central map serving as a NavCog functionality testing tool. This map visually displays the user's current position and illustrates the path taken. Overall, the navigation interface is thoughtfully designed for seamless and intuitive interactions for visually impaired NavCog App users.



(a) Planning interface (b) Navigation interface

Figure 5.2: NavCog app, user interfaces: (a) Planning and (b) Navigation [3]

NavCog's navigation mode offers visually impaired users three types of messages: Distance announcements use verbal messages or clicking sounds to indicate the proximity of upcoming actions, while action instructions guide users through turns and transitions

in their route. Point of interest descriptions offer information about landmarks and accessibility points, enhancing the user's understanding of their surroundings.

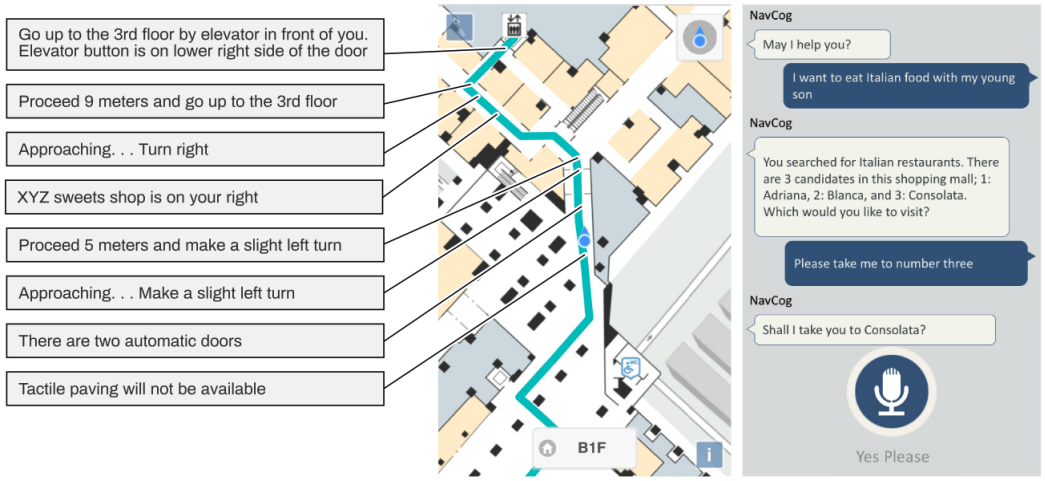
The assessment of NavCog involved six participants navigating a complex indoor environment with the app's assistance. This evaluation aimed to measure user satisfaction, effectiveness, and system usability. The findings indicated positive outcomes, underscoring the app's potential advantages for those with visual impairments. Participants expressed contentment with NavCog's guidance, perceiving it as valuable for navigation. The app's accuracy and reliability garnered favorable feedback. Additionally, the user-friendly interface contributed to a positive user experience. The study concluded that the NavCog app exhibited promising performance in aiding visually impaired users during indoor navigation tasks. Its ability to offer precise guidance, real-time localization, and audio instructions showcased its relevance in addressing indoor wayfinding challenges faced by individuals with visual impairments. Nevertheless, there is potential for enhancement. For instance, incorporating a preview mode to offer a route overview before starting could be beneficial. Furthermore, users expressed curiosity regarding potential errors after a turn, so they could recover quickly. Additionally, the satisfaction level with Points of Interest data varied, depending on their actions and personal preferences. For example, some users found the provided information unrelated to navigation and useless.

5.2.1.2 NavCog3

The NavCog project was introduced in 2015, and since then, researchers have been dedicated to enhancing the system in two primary aspects: 1. improving localization

and 2. improving navigation accuracy and user interaction. In terms of localization, the app's next version presented in [4] optimizes the localization process for efficiency and precision. The technique combines BLE beacon RSSI probability distribution estimation with pedestrian dead reckoning (PDR) using smartphone IMU data. Through this fusion, the method achieves an average localization accuracy of 0.68 meters in testing while also minimizing beacon requirements and installation effort. This makes it a practical and cost-effective solution for indoor localization. In the latest version, NavCog3, they proposed a localization technique that employs a particle filter algorithm to enhance navigation within complex indoor environments. In this method, the smartphone's motion sensors (accelerometer and gyroscope) provide data to estimate the user's movement and direction. The process begins by initializing particles across the building layout as potential user starting positions. The particle filter predicts new particle positions based on sensor data as the user moves. Bluetooth Low Energy (BLE) beacons emit signals received by the smartphone, offering proximity information to known building reference points. The particle filter compares predicted particle positions with beacon data, assigning higher weights to aligned particles and lower weights to deviating ones. Repeated iterations of this process adjust particle positions and weights, gradually improving the estimate of the user's location within the building. This approach enhances indoor navigation, particularly for visually impaired users [76, 86, 87].NavCog3, like its predecessor NavCog, uses speech as the primary method for navigation feedback. However, it goes a step further by offering on-screen information for users who rely on visual cues. The notable improvement in NavCog3 is the provision of precise turn

instructions, including advance "Approaching" notifications for upcoming turns. Users receive guidance through a combination of vibrations and sound cues to ensure they follow the correct heading effectively. Additionally, the system provides corrective guidance for route deviations and offers information about nearby landmarks and Points of Interest (POIs). Users can easily inquire about POIs through voice commands, enabling hands-free interaction, which is crucial for individuals with visual impairments. On-demand instructions allow users to access guidance by tapping the screen or using control buttons. The instructions are customized based on the user's navigation status, location, and heading (Figure 5.3).



(a) Navigation screen and examples of announcements (b) Conversation

Figure 5.3: NavCog3 App, user interfaces [86]

The effectiveness of NavCog3 was assessed through three user studies. The First Study involved 10 participants with visual impairments navigating a shopping mall with NavCog3, resulting in successful navigation and highlighting the utility of

semantic features for spatial awareness. In Study 2, 43 participants freely selected destinations within a shopping mall and used NavCog3 for navigation, demonstrating the system's capacity to confidently guide users to their desired locations. Study 3 was conducted during a 4-day conference for visually impaired individuals, showcasing NavCog3's positive impact on venue navigation and participants' independence, as evidenced by feedback from questionnaires and interviews. These studies emphasized its efficiency, which was attributed to the provision of turn-by-turn guidance, accurate localization, and inclusion of semantic features, all contributing to the improved spatial comprehension and orientation [87, 86].

5.2.1.3 VirtualNav

In recent times, researchers have been exploring alternatives to on-site assistance for blind individuals. These options include gaining prior knowledge about routes and environments through methods like spoken explanations, maps, or virtual experiences. Tactile maps and 3D models let blind individuals explore physical spaces through touch, offering accurate spatial understandings. Efforts have been directed at making these solutions more accessible, using technologies like 3D printing and interactive touchscreens [112, 38, 64, 43, 101]. A significant response to the issue of unclear maps is virtual navigation, immersing users in computer-generated environments to enhance mobility training and spatial comprehension. This incorporates sensory feedback like sound and touch [59, 62]. Smartphone-based virtual navigation taps into existing location services, allowing blind individuals access to real-world places. A virtual navigation app

has been introduced in [42], aiming to explore the usefulness of virtually acquired route knowledge in practical navigation, ultimately boosting confidence and performance in unfamiliar settings. This app is built upon NavCog3, an iOS navigation app utilizing graph-based route representations and featuring landmarks and points of interest to ensure a consistent user experience. The VirtualNav application replicates NavCog's instructions while introducing extra functionalities for simulating navigation in a virtual environment. The interface offers two navigation techniques: Virtual Leap and Virtual Walk. With Virtual Leap, users can quickly traverse route elements like turning points and landmarks, using swipes to reveal distances and instructions. Virtual Walk enables simulated step-by-step walking, including adjustable pace and directional control through swipes and phone rotation. The interface integrates more gestures for turns, stops, and location info, mirroring VoiceOver commands for user familiarity and ease of interaction.

The user study was conducted to evaluate the virtual navigation application's effectiveness in transferring route knowledge to real-world navigation for blind individuals. Fourteen participants engaged in the study by virtually learning two out of four routes across a span of three days, followed by physically navigating these routes unassisted and with an in-situ navigation tool. The study assessed the acquisition of route knowledge through virtual navigation, its evolution, and its impact on independent real-world navigation. Some participants rapidly grasped route essentials within a day, while others gradually increased their understanding of both short and long routes, forming a solid comprehension of route structures and important elements. The virtual exploration of short routes enabled most participants to transfer their route knowledge to the real

world and reach their destination unassisted. The performance of those using the virtual navigation app was comparable to those using NavCog for shorter routes. However, participants leaned more on NavCog for longer routes and less on prior virtual knowledge, resulting in minor performance improvement. The study also revealed limitations and challenges for future research. The route elements used in the study were manually annotated and static, which may not account for dynamic changes in the environment. Additionally, users encountered difficulties when they had to hold a smartphone in one hand to ensure more accurate localization while grasping their primary navigation aid with the other hand. Overall, the study demonstrated the potential of virtual navigation apps for blind individuals but highlighted the need for further improvements and research in this area.

5.2.1.4 ASSIST (Assistive Sensor Solutions for Independent and Safe Travel)

Previous studies such as GuideBeacon [21] and NavCog [3, 87, 86] have employed BLE technology for turn-by-turn navigation and location services. Google Tango, using 3D sensors and computer vision, is also investigated, with projects like ISANA [65] implementing context-aware navigation. A hybrid strategy has been adopted in the development of ASSIST [77], combining BLE for coarse localization and an augmented reality (AR) framework (initially Google Tango, now ARCore and ARKit) for precise positioning. ASSIST application consists of two main components: location recognition through hybrid sensors and map-based semantic recognition. These modules collaborate to offer users sufficient information for successful navigation and a better understanding

of their surroundings. The app does not aim to replace traditional aids used by visually impaired individuals but rather to provide additional positional and situational insights to enhance their travel experience. It utilizes floor plans to mark points of interest, calculate distances, and annotate static environmental features. The system follows a client-server structure for speed and scalability (Figure 5.4), with the app providing a multimodal interface and the server holding essential data. This design enables ASSIST to operate efficiently even in large indoor facilities and can function offline after data has been downloaded.

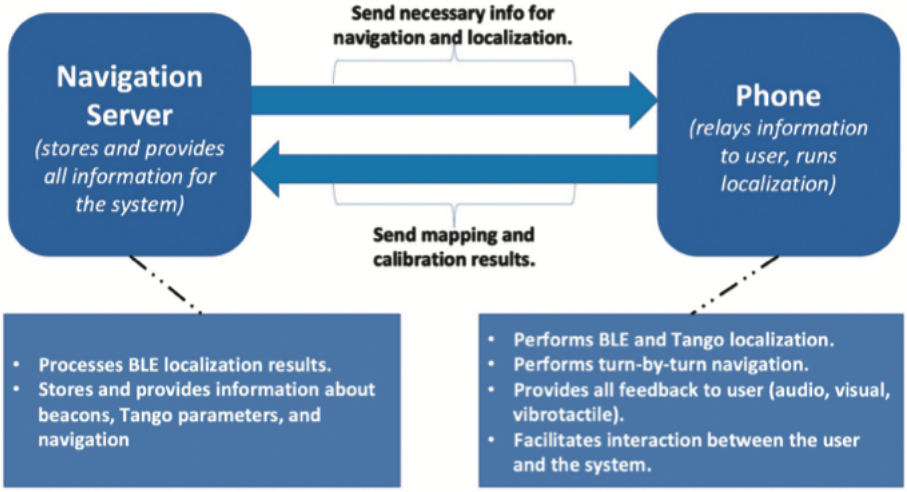


Figure 5.4: ASSIST's client-server structure. [77]

The ASSIST user interface is designed to meet the unique requirements of individuals with visual impairments, offering a range of customization options to accommodate various levels of visual impairment and navigational abilities. It consists of three main screens (Figure 5.5). The home screen serves as the starting point for users to

access navigation tools, adjust feedback settings, and personalize their experience. The navigation interface provides turn-by-turn instructions for indoor navigation, displaying essential information such as current location, upcoming navigational points, and distance to the destination, with customizable options to meet individual preferences. The voice engine interface allows users to interact with the speech engine, adjusting speech rate, volume, and language settings. ASSIST also employs a multimodal feedback system with minimal, medium, and maximal levels, catering to various user needs and preferences through a combination of auditory, vibrational, visual, and haptic cues.

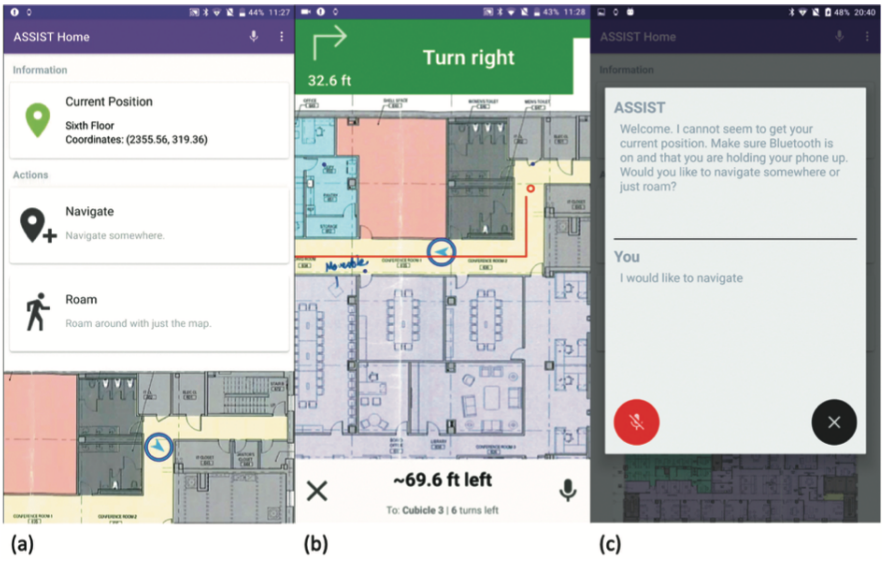


Figure 5.5: ASSIST's user interfaces: (a) Home screen interface, (b) Navigation interface, and (c) Voice engine interface. [77]

The user study assessing ASSIST included two types of tests: a usability study and a performance study. The usability study collected subjective user feedback on the app's helpfulness, safety, ease of use, and overall experience from blind and

visually impaired (BVI) participants. The performance study gathered objective data on mobility by measuring walking speed, collisions, and navigation errors for both BVI and blindfolded-sighted users. These studies utilized a Lenovo Phab 2 Pro with a built-in Google Tango 3D sensor for instructions and vibrotactile feedback. The results highlighted ASSIST's effectiveness in enhancing indoor wayfinding, emphasizing its benefits in terms of guidance, safety, and efficiency for BVI individuals. The research also stressed the importance of customized user interfaces and the potential for further enhancements in the application's functionality, providing valuable insights for developing efficient navigation solutions for individuals with visual impairments.

5.2.1.5 FIND (Friendly Indoor Navigation App for people with Disabilities)

FIND is an indoor navigation app for people with disabilities, including those with visual, hearing, cognitive, or mobility impairments and older adults. The app's primary objective is to offer a comprehensive and inclusive interface catering to a wide range of users. The study focuses on comprehending user requirements and subsequently crafting the initial interface. Three distinct interfaces are tailored for various disabilities: visually impaired, cognitive, hearing, and mobility impairments, and older adults. For example, the interface for visually impaired users utilizes smartphone corners for navigation, while individuals with cognitive, hearing, and mobility impairments, as well as older adults, benefit from a simplified design and multimodal interaction. This interface caters to the needs of users with various impairments, with a focus on customization and feedback (Figure 5.6). The study involves evaluating interfaces for inclusivity through

heuristic analysis and a focus group meeting with usability experts. Usability experts engage in various tasks and interactions; their feedback helps refine the interfaces. The research underscores the significance of creating an inclusive indoor navigation app for people with disabilities, emphasizing continuous user testing among diverse groups for improvements. This study offers valuable insights into designing and evaluating user-friendly interfaces that prioritize usability principles and user input [93].

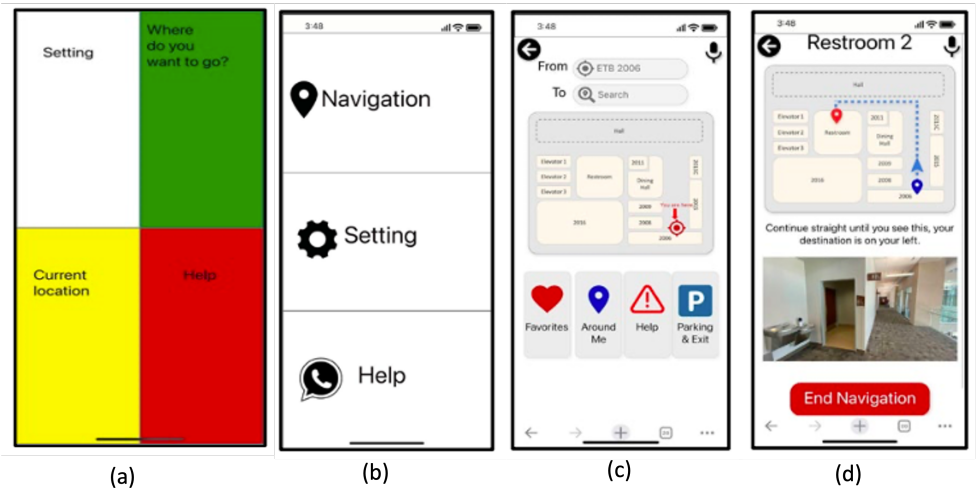


Figure 5.6: FIND's user interfaces: (a) designed for people with visual impairment, and (b, c, d) tailored for individuals with cognitive impairment ((b) main menu, (c) navigation start screen, (d) path to the restroom). [93]

5.2.2 Indoor WayFinding Apps without Maps

5.2.2.1 Easy Return: An App for Indoor Backtracking Assistance

This app addresses the limitations of existing indoor navigation technologies that often rely on infrastructure like beacons or availability of the maps. It focuses on

offering an independent indoor navigation solution for situations where infrastructure-based positioning technologies might not be available. The app is designed to run on a standard smartphone, particularly the iPhone, allowing users to carry it conveniently in their pocket without disrupting their mobility aids. Additionally, users have the option to control the app through a paired Apple Watch for seamless interaction. This approach utilizes the iPhone's inertial sensors to monitor steps and identify turns during indoor navigation. Given that many buildings have right-angle intersections in their corridors, the system concentrates on detecting $\pm 90^{\circ}$ turns. As users traverse a path, the app records left and right turns as well as step counts, creating a simplified representation of the route. The return path is essentially the inverse of the initial route. When users retrace their steps, the system compares their current location with the recorded path and delivers spoken directions based on remaining turns and step counts. The system's design is adept at handling potential issues such as step discrepancies or wrong turns, ensuring accurate and reliable guidance throughout the journey.

The user study was conducted to evaluate the effectiveness and user-friendliness of an easy return system designed for blind individuals. The study engaged six participants, all of whom were visually impaired to varying degrees, some with residual light perception. These experienced travelers were familiar with smartphones, except for one participant. Following an introduction to the system's functioning, they practiced until they were at ease with its interface and tracking mechanism. Utilizing both an iPhone and an Apple Watch, participants navigated diverse paths. The iPhone's inertial sensors tracked their steps and turns, while the Apple Watch served as the control interface. Post-trials,

participants completed a questionnaire with a Likert scale and open-ended questions. Participants found the system user-friendly and enjoyed controlling it via the Apple Watch. Yet, one expert traveler rated the system marginally useful and didn't rely heavily on it. Feedback highlighted enhancements like recognizing various turn types, marking multiple routes, and adding pause/resume capabilities. Participants envisioned real-world benefits in places like shopping malls and exploring buildings. The system's performance showed a high success rate for return paths, especially in challenging or distracting environments with three or four turns [29].

5.2.2.2 PathFinder

PathFinder is an innovative navigation system designed specifically for blind individuals to navigate unfamiliar indoor environments without relying on maps. The system uses a participatory design approach to investigate the types of useful information for blind people when navigating an unfamiliar building. The study found that intersections, directional signs, and textual signs provide the most useful information. Building on these insights, the initial mapless navigation prototype of PathFinder was developed. PathFinder is a suitcase-shaped robot equipped with a handle interface, an RGBD camera, a high-resolution smartphone camera (iPhone 12 Pro), LiDAR, and an audio feedback system to convey detection results to the user (Figure 5.7). Operating on a map-less navigation algorithm, PathFinder guides users to the subsequent intersection or hallway endpoint. It incorporates a sign recognition algorithm to read directional and textual signs, enhancing user confidence. Through iterative design and feedback from

five blind individuals, improvements were made to intersection and sign readings and the handle interface. A "Take-me-back" function was also introduced based on the user's feedback. A study with seven blind participants compared PathFinder to their usual aids, and results showed increased confidence and reduced cognitive load. While some preferred prebuilt map-based systems, they recognized PathFinder's value for various buildings, offering a balance between usability and functionality. [60].

Figure 5.7: PathFinder, a Map-less Navigation System for Blind People [60]

Many existing indoor navigation systems heavily rely on the availability of a building map. These systems employ various sensors, including BLE, WIFI, or cameras, which demand pre-installation or continuous real-time usage during navigation. Cameras, in particular, need a clear view of the environment. Given the labor-intensive nature of infrastructure installation and the challenge of handling additional devices like cameras

for individuals with visual impairments who are already holding a cane or guide dog leash, we introduce our smartphone-based (iOS) WayFinding application. This system operates with just the building map, utilizing the smartphone's built-in sensors to compute the shortest route from the current user's location to the desired destination, update the route as necessary during traversal, and provide accessible navigational support to the user as they are following the route. Furthermore, in scenarios where the building map is unavailable, we introduced the BackTracking application, building upon the work presented in [29]. This system assists individuals with visual impairments in retracing previously traveled paths (e.g., from the front entrance of a building to a specific office room), facilitating their return to the original starting point. The Backtracking app utilizes data recorded during the initial route traversal (way-in) to generate support for users when walking back (return). To achieve this, the app gradually aligns the partial return route with the original way-in route but in reverse. Notably, both of these apps do not rely on data from the smartphone camera, allowing users to conveniently keep their phones in their pockets, which is how the apps were tested in our experiment. I will explain the WayFinding application in the following sections, highlighting my primary contributions to this system. This includes insights into map and graph representation construction, two distinct localization technologies enhanced by particle filtering (completed by my colleague), interface design, and a comprehensive description of our evaluation process. The evaluation involved a user study with seven blind participants who tested both applications. Lastly, I will present the findings and results from this study.

5.3 The WayFinding System Components

5.3.1 Building the map

The WayFinding system requires the preparation of the building's map to offer navigation assistance. Similar to GPS navigation for outdoor areas, the WayFinding system focuses on providing indoor turn-by-turn navigation, aiding individuals in navigating through complex indoor layouts, such as office buildings. This assumes that the building's map is available and it constitutes a network of corridors intersecting at various angles. The map is accessed offline and employed to construct the internal graph structure utilized for navigation, incorporating details about landmarks or points of interest.

5.3.1.1 Map creation

The initial phase of map creation involves obtaining a floor plan of the environment, essentially an image illustrating the arrangement of walls and doors and integrating it into the WayFinding application. The digital building wall maps were generated by tracing the original maps using the SIM web application [104]. A developer indicated the map's waypoint (WP) and point of interest (POI) locations. For our experiment, we have selected three buildings within our UC Santa Cruz campus: Engineering Building 2 (E2), the Jack Basking Engineering Building, and the Physical Science Building. It's worth noting that our app can accommodate numerous maps, provided their floor plans are accessible.

5.3.1.2 Graph Representation

Similar to NavCog system [3], WayFinding characterizes the navigable area of the map as a graph, composed of a collection of WayPoints (WP) and route segments. WayPoints can be categorized as junction points, transition points, dead-end points, or exit doors, and route segments connect two WayPoints bidirectionally when there's a walkable straight path between them. A route connecting any two WayPoints is a sequence of interconnected route segments, which facilitates turn-by-turn navigation. This arrangement provides a series of direct segments for straightforward walking and turning, ensuring both efficiency and accuracy. In spaces where representation as one-dimensional paths is not feasible, such as open spaces lacking any reference system (e.g., where there are no walls for guidance), users might deviate from the intended route [113]. To address this, we have strategically placed WayPoints to guide users effectively.

5.3.1.3 WayPoints

The various WayPoints (WPs) on different maps are defined by their name, coordinates, and descriptions, indicating their junction type (L-junction, T-junction, X-junction, open-space, or junction), exit-door, dead-end, or corridor. Each WayPoint can serve as a starting/ending node, allowing users to initiate navigation from there or reach it as their final destination. Alternatively, it can serve as a dead-end point. If the user reaches this node and it's not their destination, they should retrace their steps until the system guides them to the correct path. The remaining WayPoints serve as junction points or nodes within open spaces.

By utilizing the optimal route generated by the system for user navigation, I've implemented code to refine WayPoint descriptions, allowing for more detailed differentiations. For instance, a door might be positioned in the middle of a corridor the user is traversing, or an L-junction could be categorized as either a left or right L-junction. At each time t, the localization algorithm (as will be discussed in Section 5.3.2) produces an estimated location $p(t)$ of the user. However, rather than directly using the 2-D locations p, we consider the *projected* locations $\bar{p}(t)$ onto their associated route segments (discussed in Section 5.3.3). This approach is justified by the nature of typical buildings with networks of corridors. To determine the direction of a specific point relative to a segment on which the user is positioned, I calculate the cross-product between vectors formed by the line's start point to the endpoint and the line's endpoint to the given point. This calculation yields three possible values: a "positive value" if the point is to the right side of the segment, a "negative value" if the point is to the left side of the segment, or "zero" if the point lies along the segment. When encountering a T-junction scenario, I utilize the slopes of the connected nodes to determine the specific type of T-junction. This involves calculating the slopes between the current WayPoint (WP) and its three connected nodes within the T-junction arrangement. Subsequently, I identify if the T-junction is rotated or not based on these calculated slopes of the connected nodes. However, in the interest of simplicity and to avoid overwhelming users with unnecessary details, we decided to provide users with a general notification of "T-junction" for both rotated and regular T-junctions.

5.3.1.4 Landmarks and Points Of Interest (POIs)

Landmarks and points of interest (POIs) play a crucial role in indoor navigation for individuals with visual impairments. These elements provide necessary reference points and contextual information that can aid blind users in navigating unfamiliar indoor environments. Landmarks are distinctive features within a building that serve as easily identifiable reference points. These can include elements like benches, couches, desks, cabinets, or unique architectural elements like pillars and alcoves. By incorporating landmarks into indoor navigation systems, blind users can receive cues that help them orient themselves and understand their surroundings. On the other hand, points of interest are specific locations within a building that hold importance for users. These could be restrooms, elevators, staircases, exit doors, specific rooms, or coffee shops. Integrating POIs into navigation systems allows blind users to receive information about the proximity and direction of these important destinations. This feature boosts their independence and confidence while traversing indoor spaces.

In the map creation process, a developer identifies the positions of landmarks and corresponding descriptions. Subsequently, I determine the nearest segment from the point as the corresponding segment for each landmark. Depending on the subject's orientation along that segment, I establish whether it lies on their right or left side to provide appropriate notifications.

5.3.1.5 Defining Routes For The Study

We have created a practice route within the E2 building to help participants become acquainted with both applications. This route is relatively straightforward, involving two turns and featuring a few landmarks along its 62.86-meter length (see Table 5.1). We deliberately removed one segment from the graph, promoting the system to determine a more complex shortest path from the established starting point to the known endpoint (see Figure 5.8).

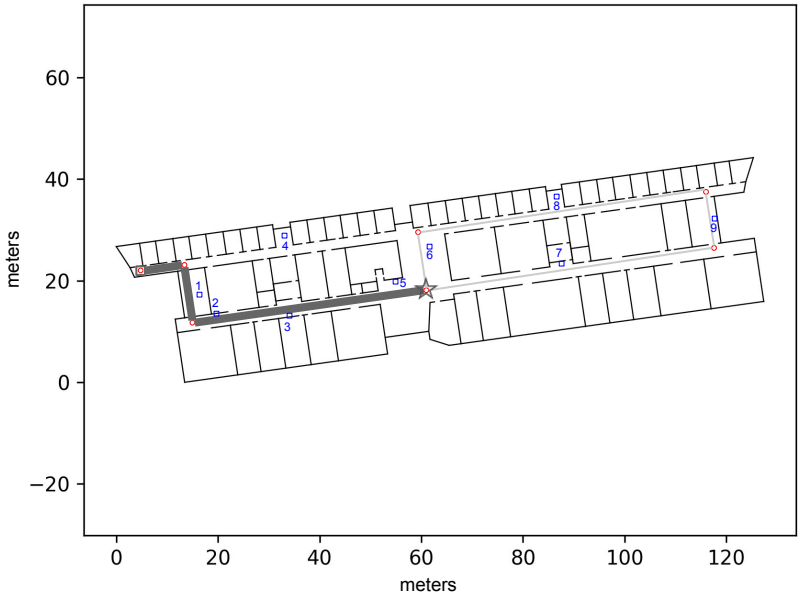


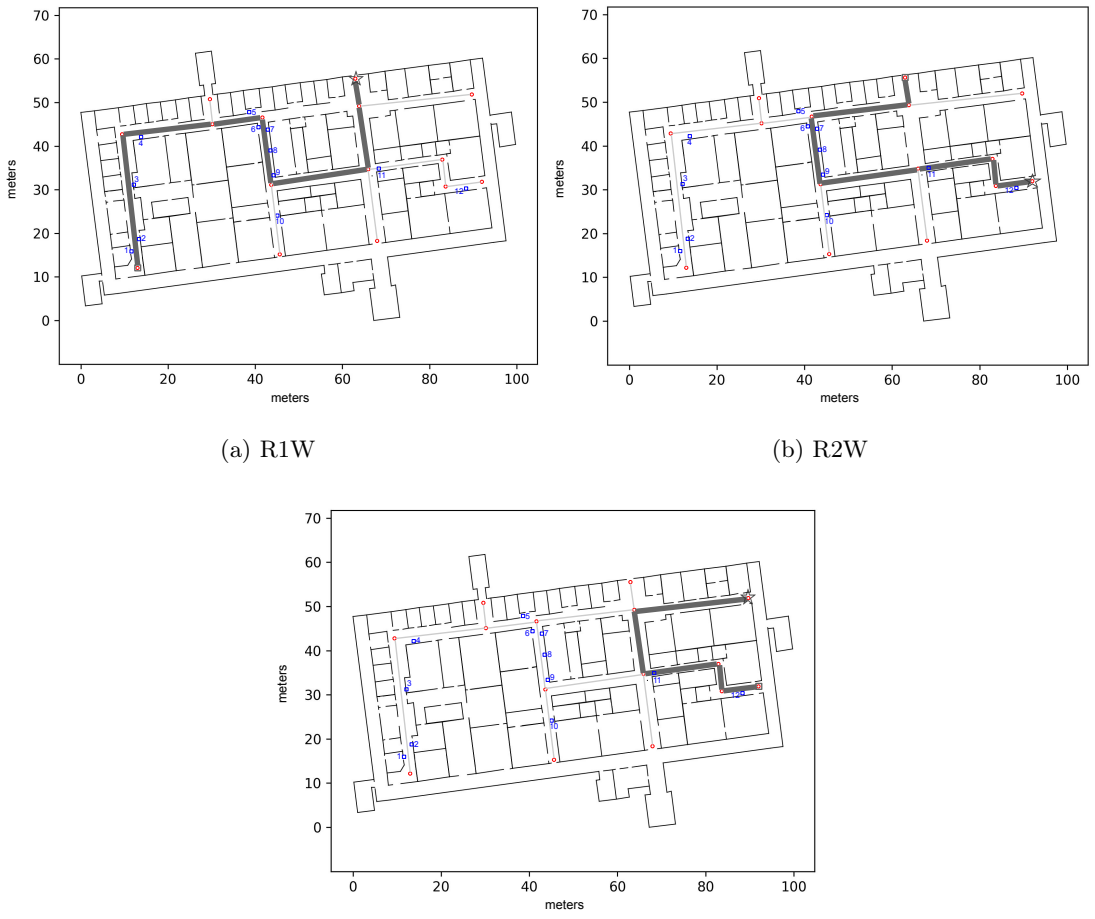
Figure 5.8: Floor plan of the E2 building. Waypoints are shown in red, and traversability graph edges are shown in gray. The start and end waypoints are marked with a square and a star, respectively. The shortest path is shown with a thick dark gray line. Landmarks are shown in blue and enumerated (see Table 5.1 for landmark listing).

Landmark ID	Landmark type
1	staircase
2	benches
3	office
4	benches
5	water fountain
6	staircase
7	benches
8	couches
9	staircase

Table 5.1: List of landmarks in E2 building.

Furthermore, we have established three routes on the second floor within the Baskin Engineering building, each designed with distinct levels of complexity and predetermined starting and ending points (see Figure 5.9). This building is situated a short distance from the E2 building. Route R1W included one L-junction, two T-junctions, and two X-junctions, with a total of 4 turns. On the other hand, R2W featured two L-junctions, two T-junctions, and two X-junctions, incorporating 5 turns. R3W comprised two L-junctions and two X-junctions, involving 4 turns. The corridor widths varied from 1.9 m to 6.0 m. The existing landmarks for these routes are detailed in Table 5.2. The building was generally quiet, with few students and researchers occasionally encountering in the corridors. All junctions of the routes were at 90 degrees. Additionally, a few potentially challenging situations were incorporated into the routes, including a narrow door (always kept open) that had to be traversed, a side staircase in a narrow corridor that had to be avoided (Figure 5.24 (c)), multiple alcoves (recessed areas) that had to be navigated (Figure 5.27 (b)), and a turn in a wide (10 m $\times$ 6 m) open space, partially

visible in Figure 5.28 (b)).



(a) R1W

(b) R2W

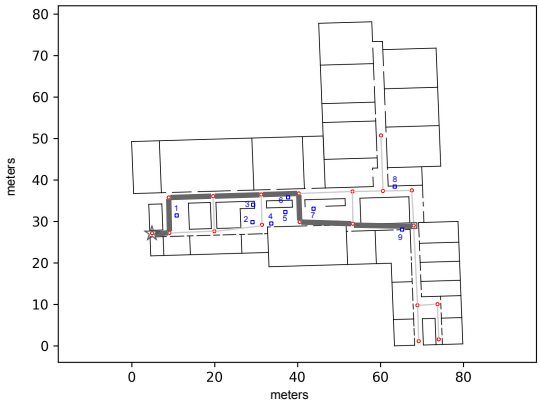
(c) R3W

Figure 5.9: Floor plan of the Baskin Engineering (BE) building. Waypoints are shown in red, and traversability graph edges are shown in gray. The start and end waypoints are marked with a square and a star, respectively. The shortest path is shown with a thick dark gray line. Landmarks are shown in blue and enumerated (see Table 5.2 for landmark listing).

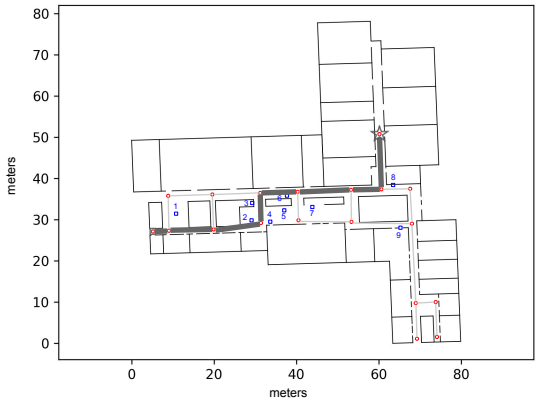
Landmark ID	Landmark type
1	alcove
2	benches
3	staircase
4	photocopiers
5	alcove
6	alcove
7	pillar
8	couches
9	pillar
10	pillar
11	door
12	cabinets

Table 5.2: List of landmarks in Baskin Engineering building.

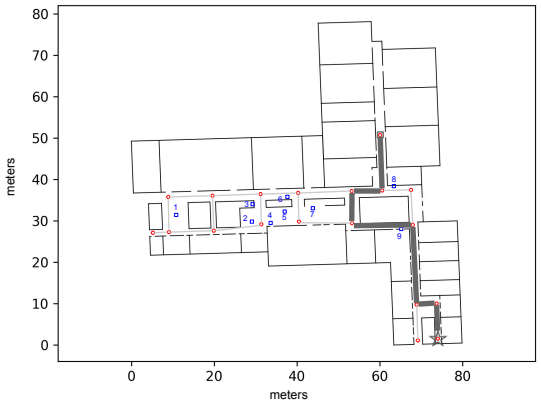
Additionally, We selected three more routes of varying complexity within the Physical Science building. These routes incorporate more open spaces to provide a higher level of challenge (see Figure 5.10 and Table 5.3 for landmark listing). The choice of the Baskin Engineering (BE) and Physical Science (PS) buildings is attributed to their proximity and ease of access from the E2 building. This decision was made to enhance convenience for visually impaired individuals. Table 5.4 provides a comprehensive overview of the specifics of each established route.



(a) R4W



(b) R5W



(c) R6W

Figure 5.10: Floor plan of the Physical Science (PS) building. Waypoints are shown in red, and traversability graph edges are shown in gray. The start and end waypoints are marked with a square and a star, respectively. The shortest path is shown with a thick dark gray line. Landmarks are shown in blue and enumerated (see Table 5.3 for landmark listing).

Landmark ID	Landmark type
1	open space with couches
2	staircase
3	staircase
4	desks
5	staircase
6	staircase
7	couches
8	coffee shop
9	elevator

Table 5.3: List of landmarks in Physical Science building.

Path-Name	Building	Path's Length (meters)	Num. of Turns	Num. of landmarks
Practice-Path	E2	62.86	2	4
R1W	BE	122.59	4	10
R2W	BE	97.04	5	6
R3W	BE	72.25	4	2
R4W	PS	77.11	4	5
R5W	PS	74.97	3	4
R6W	PS	74.67	6	1

Table 5.4: Summary of Routes' Characteristics.

5.3.2 Localization

Building on our prior work in [81], we used two distinct Pedestrian-Dead-Reckoning (PDR) algorithms, Azimuth/Steps (A/S) and RoNIN (R) [45], integrated with a particle filter for user localization and tracking (implemented by another researcher in our group). RoNIN generates velocity vectors that can be integrated into displacement vectors but face challenges in accurately estimating velocity for some users. To enhance accuracy, we introduced personalized correction factors known as RoNIN scalers, derived by dividing actual path lengths by RoNIN's estimates.

The A/S algorithm generates 2-D step vectors based on estimated length and phone orientation. Constraints were implemented to refine step detection accuracy by filtering out minor movements and reorientation using RoNIN thresholds for stationary periods and avoiding consecutive steps within $<$0.4 seconds to prevent overcounting. In our experiment, calibrated step lengths were established by dividing the path length by the step count, following similar approaches in [84, 6]. Our analysis, including applying our trained step-length calculator model to the WeAllWalk data set, showed that models trained on sighted data didn't predict step lengths accurately for blind walkers. Thus, calibration became crucial due to unique challenges faced by blind individuals, such as different gait patterns and noticeable side-to-side movements with tools like white canes [44, 49, 30, 103, 81, 69]. The particle filter further refined step length, similar to [6], enhancing accuracy in tracking blind individuals' movements.

Impenetrable walls on building floor maps notably enhance localization accuracy when available [120, 32]. Particle filtering [102] represents the walker's location using statistics like the mean location. Each particle holds attributes like position, drift angle, and step length (for A/S only), adjusted at each detected step. Weight adjustments occur based on criteria, reducing weights for crossing walls, significant deviations from the average position, or room entries, minimizing localization uncertainties.

The A/S and RoNIN algorithms utilize smartphone inertial sensors. While both ran simultaneously, only one guided the user, serving dual purposes: a backup in case of failure (required once during the experiment) and enabling comparative analysis of localization data. They position users within a fixed "world" frame, aligning the Z-axis

with gravity. Calibration aligned this frame with the floor plan's frame. By assuming known starting points and directions, we measured the angle between the reconstructed path and the intended direction after 6 steps.

5.3.3 Segment Assignment

At every time step, whether it's A/S with PF or RoNIN with PF, the localization algorithm calculates the user's estimated location as $p(t)$. This data guides the app to create a route to the destination and issue relevant notifications. Rather than using the raw 2-D locations p, we use the projected locations $\bar{p}(t)$ onto the route segments, a technique implemented by my colleague in our lab.

As users walk, we can safely project their location onto a corridor's route segment. Yet, at junctions, there's a risk of localization errors leading to incorrect assignments. To handle this, we implemented a mechanism: As long as a walker's projected location $\bar{p}$ stays more than a set threshold T (in our case, 1.5 m) from any intersecting junction, they stay associated with a route segment. When $\bar{p}$ gets closer than this threshold to a junction, the association with any segment becomes ambiguous, disconnecting p from all segments (see blue arrows in Figure 5.11). Reassociating happens when p projects onto a segment reaching a distance of T or more from the junction, establishing a new segment association (see Figure 5.11).

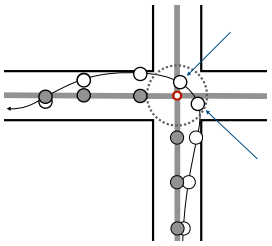


Figure 5.11: Segment Assignment Example: The walker's path is illustrated using white circles, while their projection onto assigned segments is denoted by grey circles. Segment assignment is paused when the walker enters a circular area centered at the junction (marked by the red circle) with a radius of T.

5.3.4 Routing

The WayFinding system utilizes a graph-based environment for routing, representing walkable areas as WayPoints connected by line segments (edges). Each edge's weight corresponds to the distance between its connecting points. The initial and final points are predefined for all routes described in Section 5.3.1.5. While the system can accommodate various starting and ending nodes, we intentionally selected these points for consistency among subjects and to establish standardized routes. Our primary objective is to determine the shortest path, whether from the initial starting point (before walking begins) or from the user's current location to the intended destination.

I utilized Apple's GameplayKit toolkit for iOS development[1]. While primarily designed for game-related tasks such as creating intelligent behaviors, pathfinding, and simulations, it has found applications in solving pathfinding and graph-related challenges

[1]https://developer.apple.com/documentation/gameplaykit

in non-gaming apps. To set up the graph, I created nodes based on predetermined WayPoints and their respective positions. I established two-way connections between nodes using edges, as subjects can move in both directions on the edges when a direct path exists. The edge's cost is equivalent to the Euclidean distance between the connected points. To find the shortest path to the destination from the current position, I introduced a new node representing the current position along its corresponding edge, connecting it to the two existing nodes initially linked by the edge. I used the FindPath function to retrieve the optimal path sequence from the current position to the destination, determined by the Dijkstra algorithm. Dijkstra's algorithm is widely employed for finding the shortest path in a weighted graph. It starts from an initial node, systematically explores neighboring nodes with the smallest known distances, and maintains sets of visited and unvisited nodes. The algorithm selects nodes with the smallest known distance, updates neighbor distances if shorter paths are found, and is effective for non-negative edge weights. In cases where all edge weights are non-negative, it guarantees the shortest path discovery.

5.3.5 Turn-By-Turn Instructions

The WayFinding system offers turn-by-turn directions through speech instructions, facilitating hands-free engagement. Similar to [86], the WayFinding system visually displays essential navigation information on the screen. This includes the map, a representation of the graph featuring the shortest path highlighted, marked WayPoints and landmarks, the user's current position, and provided instructions. This dual approach caters to users with varying degrees of visual ability: those who prefer auditory guidance

and those who can benefit from visual cues. Moreover, the visual information serves the purpose of application testing and monitoring (see Figure 5.12).

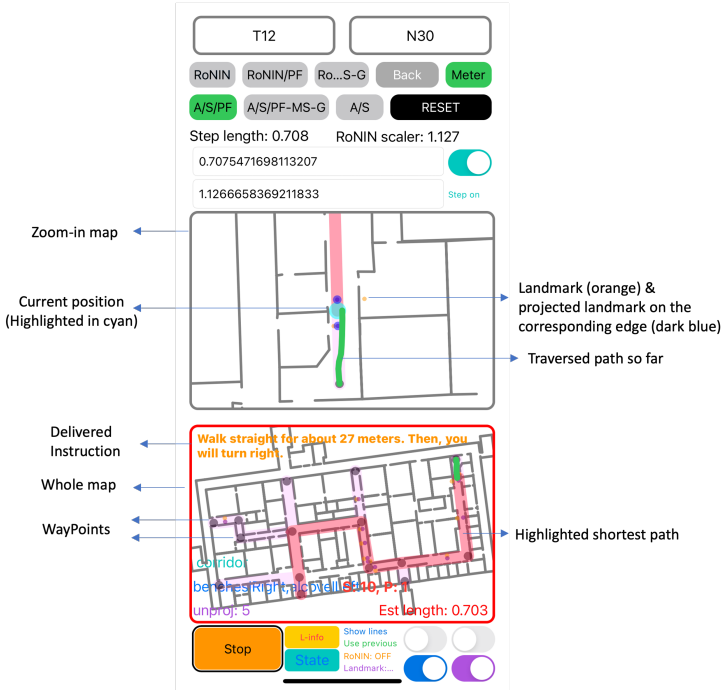


Figure 5.12: Sample screenshot displaying the information provided by the WayFinding application.

The provision of accurate instructions for users to reach their destinations relies on real-time location tracking. It's crucial to acknowledge that, despite our efforts to minimize errors using particle filters, there may still be location inaccuracies of up to 2-3 meters. Disregarding these errors when sending notifications can lead to significant issues. For example, the app might send a notification right before it's completely certain about the user's location at a junction. This situation is shown in Figure 5.13, where the user's actual location might be slightly in front of or behind the junction, potentially

causing confusion about the correct turn. To overcome potential location inaccuracies, our purposeful design approach entails providing advance notifications as users approach a junction. This proactive strategy ensures that users haven't reached the junction by the time they receive the notification. While the concept is straightforward, its viability was not immediately evident. Before the experiments, users are informed about this approach and encouraged to start searching for the next available opening immediately upon hearing the notification, either with their cane or dog. A key objective of our experiments was to confirm the success of our participants in this task and to assess their acceptance of this strategy. Extensive pre-experiment testing has validated the effectiveness of this notification approach when the app detects the user is approximately 7 meters away from the next WayPoint. This approach also considers the common behavior of individuals who continue walking while listening to notifications and may advance by 2-3 meters before fully processing the information.

Collaborating with another researcher in our lab, we've defined distinct states with different priorities. These states are governed by a state transition mechanism that enables us to create accurate notifications to guide users through each step of their navigation journey. The specific state is determined by the user's current route and position, focusing on factors such as the user's location relative to the next WayPoint in the route and nearby landmarks.

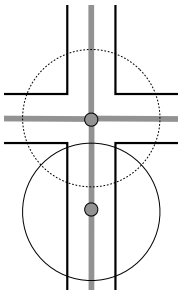


Figure 5.13: Hypothetical junction scenario: The walker's estimated location is denoted by the gray-filled circles. The larger circles illustrate the expected radius of location uncertainty. If the app identifies the walker at the junction center (top circle), the actual location might fall anywhere within the dashed circle of uncertainty, which could be before or after the junction. If the app places the walker at the lower circle, it confirms that the walker's actual location is indeed before the junction.

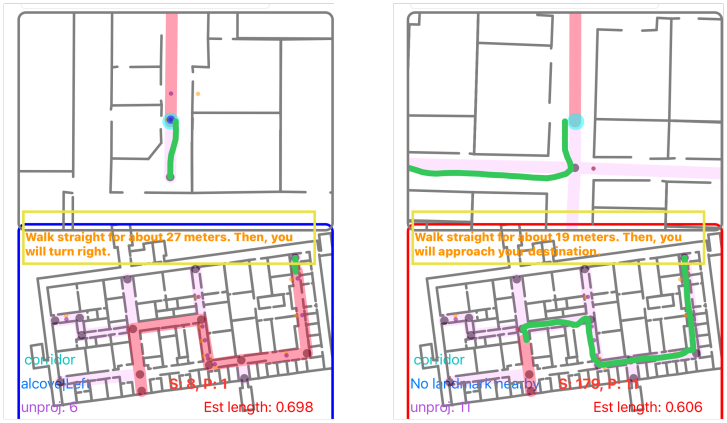
5.3.5.1 Notification Generation States

We are considering various states that the user may encounter. Note that in case of conflict where a new state is triggered while a prior one is active, the ongoing state continues except for the (S3) type state alerting the user of an upcoming turn, as it takes priority due to its time-critical nature. It's also essential to emphasize that we avoid repeating the same notification, except for the (SW) type state. Additionally, we prioritize designing navigation notifications to reduce cognitive load and minimize distractions by delivering concise and short synthesized speech sentences.

- S1: The S1 state is initiated when the user enters a new route segment. The specific content of the notification generated in the S1 state is determined by the

upcoming WayPoint and the distance to that particular point. Here are examples of notifications that may be issued in the S1 state:

- The next WayPoint is a junction: "Walk straight for about XX [meters/feet/steps]. Then, you will turn [left/right]." (see Figure 5.14 (a))
- The next WayPoint is the destination point: "Walk straight for about XX [meters/feet/steps]. Then, you will approach your destination." (see Figure 5.14 (b))

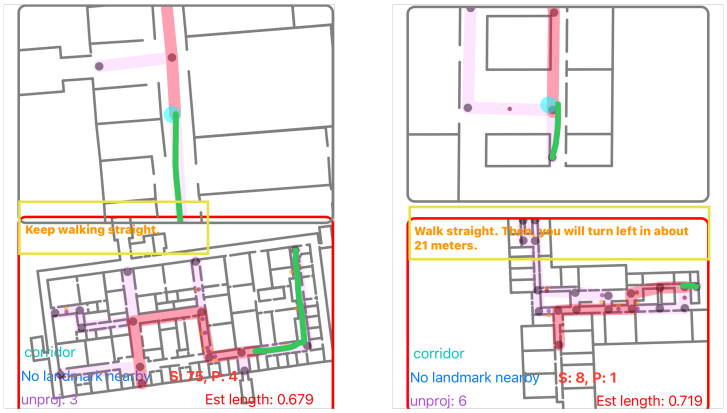


(a) (b)

Figure 5.14: Notifications triggered when the user enters a new segment, denoted by the highlighted yellow box. (a) When the upcoming WayPoint is a junction. (b) When the upcoming WayPoint is a destination point. The pink lines indicate the optimal route, while the green lines track the user's actual path. The cyan dot represents the user's projected location on the associated segment.

- S2: This state activates when the user is within 7 meters of a Non-Turning Junction

(NTJ), where they should continue straight. Its purpose is to guide the user correctly when they encounter this junction. In the S2 state, the notification message "Keep walking straight" ensures the user proceeds through the upcoming junction without making unnecessary turns. If both S1 and S2 states occur at the same time, the instructions provided combine aspects of both to offer comprehensive guidance to the user (see Figure 5.15).



(a) (b)

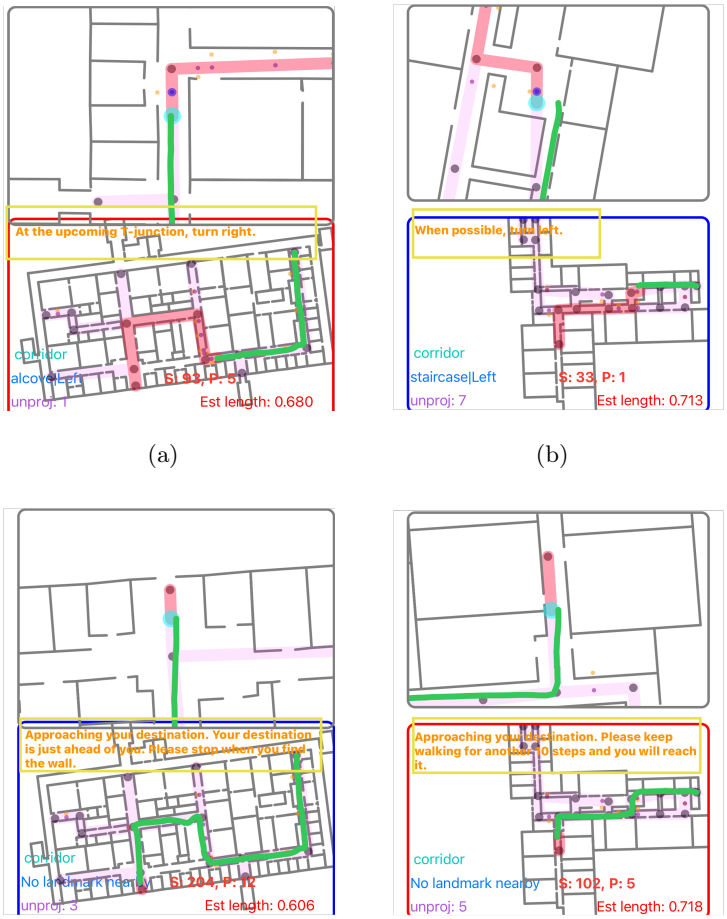
Figure 5.15: Notifications for passing through Non-Turning Junctions (NTJs), highlighted within the yellow box for S2. (a) Activation of the S2 state, and (b) Activation of both the S1 and S2 states. The pink lines indicate the optimal route, while the green lines track the user's actual path. The cyan dot represents the user's projected location on the associated segment.

- S3: The S3 state activates when the user is within 7 meters of a Turning Junction (TJ), an open space, or a destination point. It generates precise instructions to

confidently guide the user through the upcoming turn or open space, based on the nature of the next WayPoint (refer to Figure 5.16):

  - The next WayPoint is a Turning Junction: ”At the upcoming [X, left/right L, T] junction, turn [left, right].”
  - The next WayPoint is an open space: ”When possible, turn [left, right]”
  - The next WayPoint is a destination point: ”Approaching your destination. Your destination is just ahead of you.” If the route ends at a wall or a closed door, this notification is followed by: ”Please stop when you find a wall.”

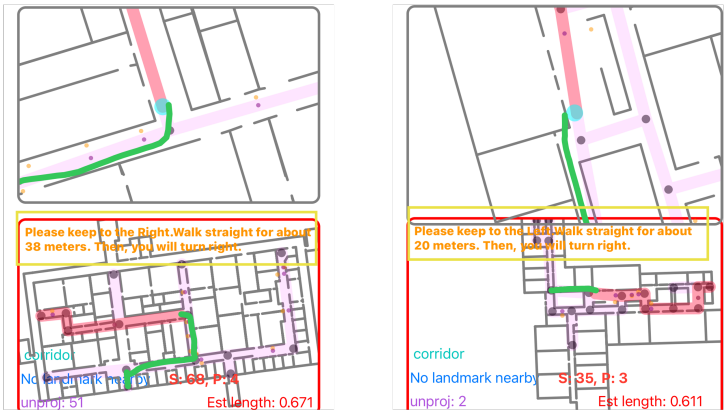
- S4: State 4 activates when the user enters a segment that requires them to "keep left" or "keep right". These segments are pre-defined and labeled for precise notification generation. The purpose of State 4 is to provide clear instructions, ensuring the user remains on the correct side of the segment. This may involve avoiding potential hazards like emergency showers or guiding the user through extensive corridors and open spaces (see Figure 5.17). Both states 1 and 4 trigger upon entering a new segment, so the S4 notification incorporates the S1 instruction.



(a) (b)

(c) (d)

Figure 5.16: Notifications for the S3 state when the user is in the proximity of a turning point: (a) at an approaching junction or (b) at an upcoming open space. Additionally, scenarios depict the user's proximity to the endpoint: (c) approaching a wall signifying the destination and (d) positioned within a corridor. The pink lines indicate the optimal route, while the green lines track the user's actual path. The cyan dot represents the user's projected location on the associated segment.



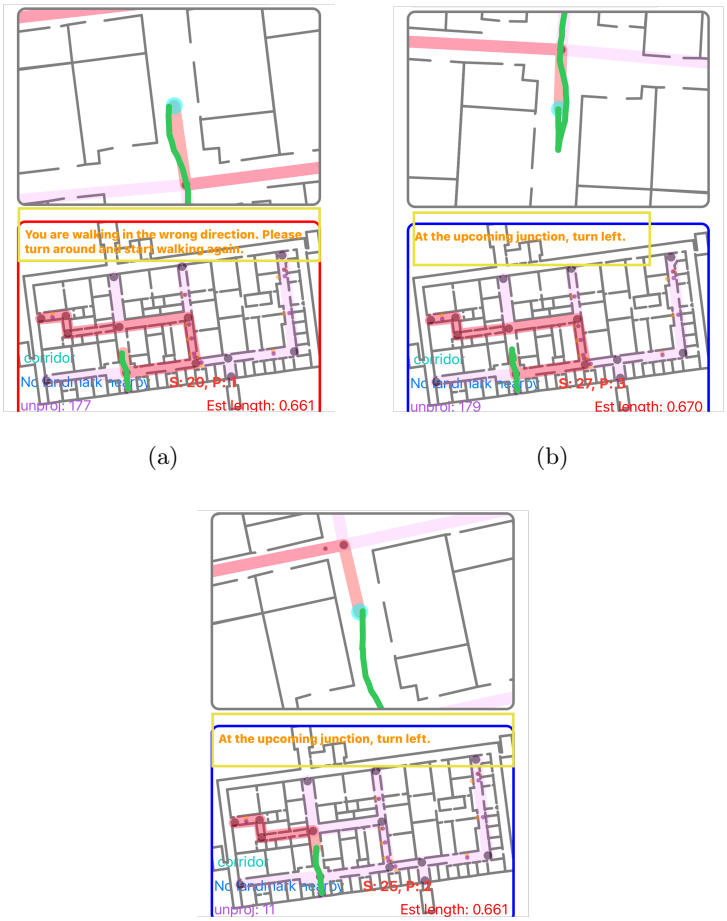
(a) (b)

Figure 5.17: Notifications for situations where the user must keep left or right, highlighted in the yellow box for S1+S4. (a) S4 is triggered because of emergency showers on the left side, and (b) S4 is activated to guide the user to stay on the correct side of the segment while navigating through a wide open space. The pink lines indicate the optimal route, while the green lines track the user's actual path. The cyan dot represents the user's projected location on the associated segment.

- SW (Wrong Direction): The SW state, or "Wrong Direction" state, activates when the user has been consistently walking in the wrong direction for at least 4.5 meters. The chosen length of 4.5 m was determined through trial and error in initial experiments and aligns with the anticipated radius of localization uncertainty. This state's purpose is to alert the user to their incorrect path and prompt them to turn around (see Figure 5.18 (a)). This helps prevent the user from deviating further from the correct route and guides them back on track. If the user continues in the

wrong direction for another 4.5 meters, the same notification is repeated. While walking in the wrong direction, other states cannot be triggered. However, if the user corrects their path and starts moving in the right direction, they transition to the appropriate state, S1, triggering the relevant notification (see Figure 5.18 (b)). Additionally, the system consistently aims to find the optimal route from the user's current location to the destination. If walking in the wrong direction reveals a new, more efficient path to the destination, the system guides the user to follow this newfound route (see Figure 5.18 (c)).

- S5: The S5 state activates when recognized pre-labeled landmarks are within 2 meters, serving as navigational references. Unlike approaching junctions, the shorter distance threshold here is justified by the lower need for advanced notice for nearby landmarks. Notifications inform users about these nearby landmarks, detailing their location, type, and their left/right positioning concerning the user's walking direction (see Figures 5.19 (a) and (b)). Multiple landmarks within range prompt notifications for each. This information enhances spatial awareness and informs route decisions. These notifications hold the lowest priority and can only be activated when no other states are active (Figure 5.19 (c)).



(a) (b)

(c)

Figure 5.18: (a) The SW state is triggered when the user misses a turn and continues walking straight. The system issues a notification to prompt the user to turn around. (b) The user follows the notification and changes direction, successfully returning to the correct route. (c) Despite the user's continued wrong direction, the system adapts by identifying a new optimal path and guiding them through the alternative route towards the destination.

Figure 5.19: (a) The system notifies the user about a single nearby landmark. (b) The user receives notifications about all nearby landmarks in proximity. (c) When landmarks are located at the start of the segment, and S1 is triggered, the S1 notification takes priority over the S5 notification.

5.4 User Interface and Interaction

The WayFinding application operates in conjunction with two devices: an iPhone carried in the participants' pocket, and an Apple Watch paired with the iPhone. The iPhone uses its inertial sensors for step tracking, turn detection (left or right), velocity predictions via RoNIN, localization, routing algorithms, and generating guidance messages. Participants used the Apple Watch for system control (initiating and concluding navigation) and accessing information (requesting updated directions on demand). It's important to note that while the Apple Watch's usage is optional, it provides participants with full control over the experiment, eliminating the need to hold the phone in hand or seek assistance from experimenters. The WayFinding phone application also includes additional control keys for experimenter management throughout the study.

5.4.1 Watch Gestures for Controlling the System

Users have complete control over the system through the Apple Watch interface. Prior to starting navigation, users can choose a route from a list. The list can be traversed in both directions by swiping left or right on the Watch's face, with the route's name audibly announced by VoiceOver for easy selection. As previously mentioned, route selection also determines the starting and destination points for consistency among users (see Figure 5.20). Once a route is chosen, navigation starts by rotating the crown in either direction until hearing "Please start walking". At any point, users can halt navigation by turning the crown until the announcement "process done" is made. The watch also

provides auditory and haptic cues, including a "ding" sound, to indicate the initiation and conclusion of navigation. This feature helps users easily identify the start and end of their journey.

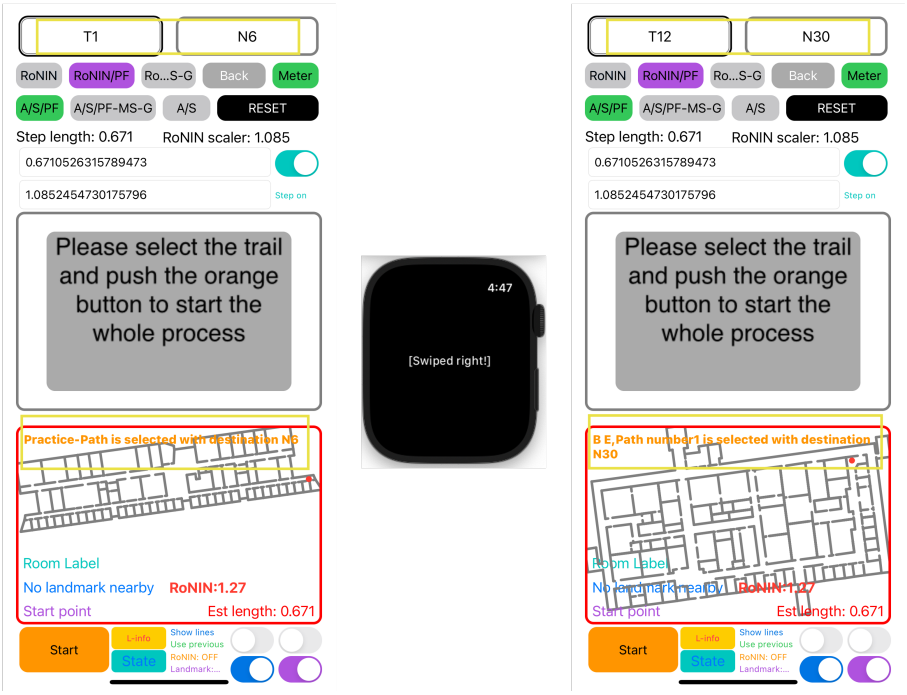


Figure 5.20: The route and ending WayPoint are changed once the user swipes right on the watch. The name of the route is announced and highlighted within a yellow box.

Additional control buttons in the phone version of the app offer experimenters customization options, including:

1. Enabling or disabling step detection beeping sounds.

2. Selecting the format for navigation instructions: meters, feet, or steps. When the step-based option is chosen, the distance can be converted into the corresponding number of steps using the user's calibrated step length.

3. Toggling the activation or deactivation of the calibration stage and the RoNIN algorithm. Both are initially turned off.

4. Selecting whether to display the traveled path as a line, allowing control over how the experiment's progress is represented.

5.4.2 Watch Gestures for On-Demand Navigation Instructions

After initiating navigation, users can perform "swipe left" and "swipe right" gestures on the watch's face to access different information. Swiping left provides a comprehensive summary of the entire path in terms of route segments and turns. This gesture offers an overview of the route from the starting point to the destination. At any point during navigation, users can request complete path instructions, which the system tailors to their current location, guiding them through the remaining part of the optimal path to their destination (see Figure 5.21). The "swipe right" gesture on the watch serves to reiterate the most recent active state instruction, taking into account the subject's current location. It's important to note that this action does not replicate the S5 instructions related to landmarks or POIs; rather, it prioritizes the navigation-related instructions.

(a) (b)

Figure 5.21: Notifications for the "swipe left" gesture on the watch during navigation. (a) The complete path is generated based on the optimal route from the starting point to the destination, marked by a highlighted yellow box. (b) When on the route, performing a "swipe left" gesture generates navigation instructions for the full path from the current position to the destination point, also marked with a highlighted yellow box.

An additional watch gesture is designated for developer use, intended to be employed only in rare and exceptional situations. In instances where a user has deviated from the intended route and has extensively explored the surrounding area without successfully retracing their path, the experimenter can assist them by guiding them to the nearest WayPoint. This involves repositioning the user within the application and,

if deemed necessary, switching between Localization algorithms such as RoNIN+PF or A/S+PF. By holding down the watch screen for two seconds, developers enter administration mode. Developers can navigate through WayPoints by swiping left or right on the watch. Switching the localization algorithm to RoNIN is achieved with an upward swipe, and a downward swipe allows the developer to reset the user's position. Once the user resumes walking, the system generates precise navigation instructions based on the reset position to ensure accurate guidance.

5.5 Experiment

Before conducting the user study with blind participants, we performed tests with blindfolded sighted individuals who used the app in combination with a long white cane for navigation. These initial tests were aimed at identifying potential app issues, including landmark positioning and determining the optimal number of landmarks to avoid overwhelming users with information. Additionally, we used feedback from these tests to refine the speech instructions provided at various route segments. Once we were satisfied that both the WayFinding and BackTracking apps functioned effectively for sighted individuals, we proceeded to invite blind participants for the actual user study.

During testing, participants initially followed three routes within the BE building using the WayFinding app. They traversed route R1W (Figure 5.9 (a); length: 123m), followed by route R2W (Figure 5.9 (b); length: 97m), and concluded with route R3W (Figure 5.9 (c); length: 72m). It's worth noting that each route's endpoint corresponded

to the starting point of the subsequent route, and all routes terminated at a closed exit door location. When testing the Backtracking app, participants retraced the same routes in the reverse order, with reversed directions. These routes were labeled as R1B, R2B, and R3B. In practice, each participant first walked R1W, R2W, and R3W using the WayFinding app, while the Backtracking app (installed on a different smartphone) collected way-in data for each route. Subsequently, participants tested the Backtracking app by traversing R3B, R2B, and R1B

Initially, we planned to include three routes within the Physical Science building as well. However, after the first two participants completed the tests in the BE building, they were already exhausted. Therefore, we made the decision to exclude the routes in the Physical Science building from this user study.

5.5.1 Population

Seven participants took part in this experiment, and you can refer to Table 5.5 for an overview of their characteristics. All participants were blind, with the majority having some residual light perception. They were experienced independent walkers. Notably, P6 was in the process of transitioning from using a guide dog to using a long cane and relearning cane usage. Additionally, P5 had a hearing impairment and used hearing aids. In terms of mobile devices, all participants, except for P7, used iPhones; P7 used a cell phone with a physical keypad. It's worth noting that only P1 frequently used a smartwatch (Apple Watch). Table 5.5 also includes the calibrated step length values obtained during the initial calibration phase, along with the average step length

derived from the particles at the end of the trials (note that step length was introduced as a state in the Particle Filter starting from P4). Note that the step lengths obtained from the particles at the end of the trials were either smaller than or equal to the step lengths measured during calibration for the same participant. As mentioned earlier, this can be attributed to the easy, straight corridor used for calibration, where users could confidently walk with longer steps.

user	Gender	Age	Blindness	Mobility aid	Step length (cm) (calibration)	Step length (cm) (final)	RoNIN multiplier	Preferred units
P1	F	73	L	Dog	48	–	0.96	Steps
P2	M	69	B	Cane	51	–	1.08	Feet
P3	M	53	B	Cane	54	–	1.14	Feet
P4	F	69	B	Cane	51	44	1.0	Feet
P5	M	75	L	Cane	44	41	1.21	Meters
P6	F	76	L	Cane	40	40	1.11	Steps
P7	F	72	L	Dog	63	58	1.08	Feet

Table 5.5: Characteristics of the participants in our study. In the "Blindness" column, "B" denotes blindness from birth, while "L" signifies later onset. The "Step length (final)" column represents the average step length value calculated at the end of the trials by averaging values. Please note that the particle filters included step length as a state starting from P4.

5.5.2 Experimental Settings

After obtaining their consent, we provided each participant with a detailed explanation of both apps and their functionalities. We encouraged them to ask questions if anything was unclear. We emphasized the importance of notifications regarding upcoming turns, assuring participants that they would receive advance notice. We clarified that, upon receiving such notifications, they should be prepared to identify the first available

opportunity to make the turn, whether it's nearby or a few meters down the path.

The initial step in our application testing process is the calibration of two critical parameters: the user's step length and the RoNIN scaler factor. This calibration process is vital to ensure precise distance measurements. We conduct this calibration in a designated corridor within the E2 building, conveniently located near our office, where participants are welcomed. During the calibration phase, we maintain a consistent path length of approximately 38.25 meters. Participants are instructed to walk this predefined path while counting their steps by the app. By dividing the path length by the recorded step count, we determine the user's specific step length. Simultaneously, we apply regression to calculate the RoNIN multiplier. This is achieved by dividing the actual path length by the length calculated by RoNIN, as explained in Section 5.3.2.

Subsequently, we accompanied participants to the practice trial's starting location. Each participant was equipped with two iPhones placed in their pants pockets, and we had previously requested that they wear pants with pockets on the day of the experiment. These iPhones served different roles: the iPhone 12 ran the WayFinding app, while the iPhone XR recorded way-in data for the initial WayFinding trials. Additionally, participants wore a wireless bone conduction headset (Shokz OpenRun) to receive app notifications, and they had an Apple Watch Series 8 to interact with the apps. We ensured that the VoiceOver settings, including speed and sound volume, were adjusted to their preference before commencing the practice trial. Furthermore, we inquired about their preferred units for receiving directions, such as meters, feet, or steps, and configured the apps' parameters accordingly (see Table 5.5). Before and during the practice trial,

we encouraged participants to become familiar with the watch's swipe left and right gestures and their various functionalities both before and after navigation started. All participants quickly adapted to these gestures, although there was initial confusion for P2, who initially interpreted "right" or "left" as referring to the axis of his left arm (where he wore the Watch), i.e., in a direction orthogonal to the forearm rather than parallel to it. Participants navigated the practice route using the WayFinding app, and upon reaching their destination, they retraced the route in reverse using the BackTracking app.

Participants were provided with the option to disable the sound of detected steps. However, all participants chose to keep the step sound enabled, with some mentioning that it reassured them about the system's proper functioning. We also asked if participants wanted to receive notifications about nearby landmarks (a few landmarks were announced during the practice trial). All participants decided to continue receiving landmark notifications. Initially, we had the landmark option disabled for P1, thinking it might overwhelm participants with excessive information. However, following feedback from P1, we decided to include this option in the app and let participants make the choice.

After completing the practice path, participants were guided to the Baskin Engineering building, where the actual experiment took place, specifically starting at the initial point of the first route (R1W). The routes are described in Section 5.3.1.5. At the beginning of each trial, participants were led to the route's starting point and properly oriented. Following this, they were reminded of the instructions on how to choose the next route from the list by swiping left and right on the watch's face. After selecting the route,

they were instructed to rotate the Watch's crown to launch the app (Figure 5.22 (a)). Additionally, participants were asked to swipe left on the Watch to receive an overview of the entire route (Figure 5.22 (b)). With these instructions, participants began walking. Upon reaching the destination, they stopped the app with the Watch's crown and had the option to take a break. Afterward, participants were repositioned at the starting point of the next route, which coincided with the endpoint of the previous route. They were carefully oriented before beginning the next route. Experimenters maintained a safe distance from participants throughout the trials to avoid influencing their routing decisions.

After completing their traversal of the final route, R3W, using the WayFinding app, participants were asked to retrace the three routes in reverse order using the BackTracking app. During this transition, participants received new headsets and Watches, which were the same models as the ones they had been using. They were instructed to replace their old devices with these new ones since both devices could be paired with only one iPhone at a time. This switch was necessary because the Backtracking app ran on a different iPhone than the WayFinding app. Additionally, we equipped participants with foot sensors attached to their feet to record foot data. Another phone was used to collect the inertial data from the phone, which was synchronized with the foot sensors. This data was utilized in a separate project related to step length prediction with blind users (results are presented in Chapter 4, Section 4.5.2).

(a) (b)

Figure 5.22: Participants interacting with the WayFinding app via Apple Watch: (a) Participant P6 initiates the app by rotating the crown at the initial point of R1w. (b) Participant P7 swipes left after starting the app to hear a complete preview of Route R3w.

To ensure the apps' performance during the trials, we employed screen mirroring through the ApowerMirror app to display the iPhone's screen on the smartphones carried by the experimenters. Given that our apps were intended to run in the foreground, we utilized the Guided Access utility found in iOS Accessibility settings. This approach helped prevent unintended button presses or gestures when the phones were stored in the participants' pockets.

Upon completing their last trial, which involved traversing route R1B with the

Backtracking app, participants and experimenters returned to the initial building. At this point, participants were invited to take part in a questionnaire that included the ten System Usability Scale (SUS) questions and several open-ended questions.

5.5.3 Observations and Results

Table 5.6 presents the duration of successfully completed routes, and indicates instances where administrative resets were necessary, or participants received verbal guidance from the experimenters. It also distinguishes trials in which participants missed turns but were able to reorient themselves with the assistance of the app's instructions.

We activated the administrative mode for route R1W for participants P1, P2, and P3, as denoted by the "R" marker in the table. In the case of P1, we switched the localization algorithm from A/S to RoNIN after encountering issues with A/S accurately tracking the participant (Figure 5.24 (a), blue line). As for P2 and P3, we performed manual resets to position them at the nearest WayPoint within the app. For all three users, the remainder of the routes were completed without any issues (except for P1 in R2W, as will be discussed later).

It's essential to highlight that, for the first three participants, the step length value (s_i) was not yet integrated into our particle filter implementation (Section 5.3.2). Consequently, A/S relied solely on the step length measured during calibration. Route R1W is among the most challenging routes in our study, primarily due to the initial segment. This segment is characterized by its considerable length, exceeding 40 meters, as well as its narrow width and the presence of several obstacles, including benches and

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

Table 5.6: Summary of WayFinding Route Experiments. Reported durations (in seconds) for successfully completed routes. Gray background indicates participants who missed turns but completed the route with app guidance. R denotes a system reset requirement, while E indicates route completion with verbal input from the experimenter.

staircases located on the right side. Participants received explicit instructions not to climb the stairs when encountering them (Figure 5.24 (c). This caution led participants to take substantially shorter strides compared to their calibration settings. By the time they reached the corridor's end, the accumulated length error had become significant, making tracking challenging, even with particle filtering. For P1, we successfully addressed the issue by transitioning to RoNIN, which resulted in excellent performance (Figure 5.24 (a), red line). However, it's worth noting that RoNIN faced occasional challenges in the same area, as it couldn't effectively track P2 ((Figure 5.24 (b), red line)). After we incorporated step length values into the particle filtering process, this problem ceased to occur. Consequently, we were able to utilize A/S with success for all other participants (see Figure 5.23).

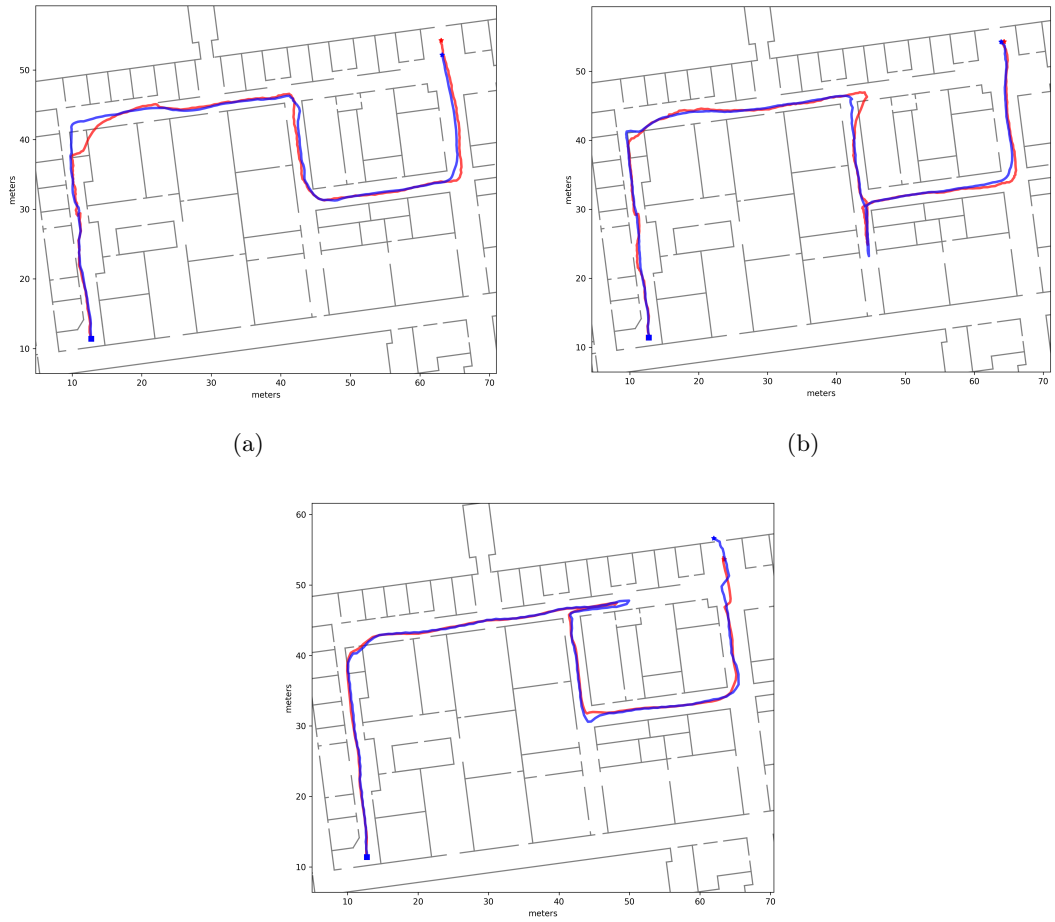
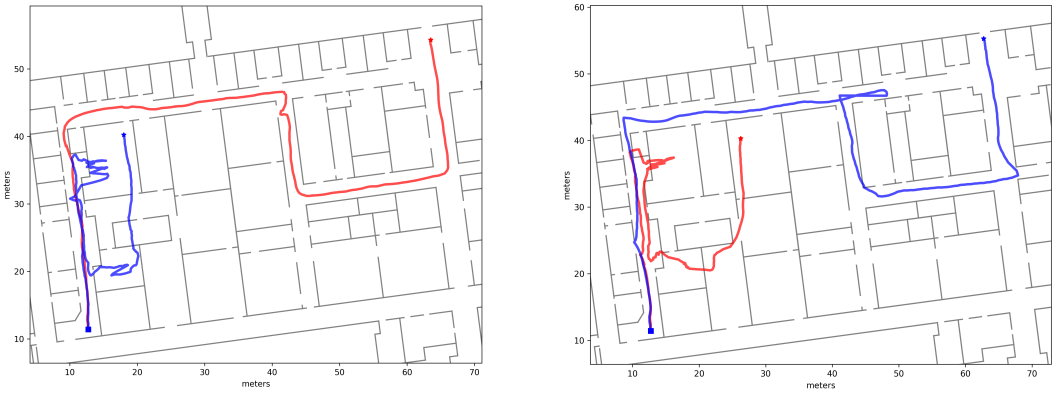
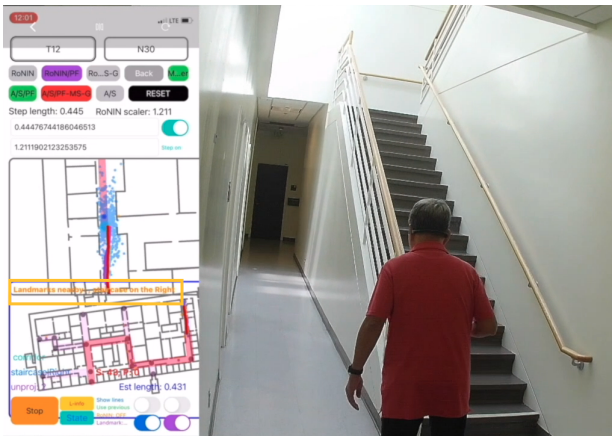


Figure 5.23: (a)–(c) Recorded route R1W using A/S (blue line) and RoNIN (red line) for (a): P4, (b): P5, and (c): P7.



(a) (b)



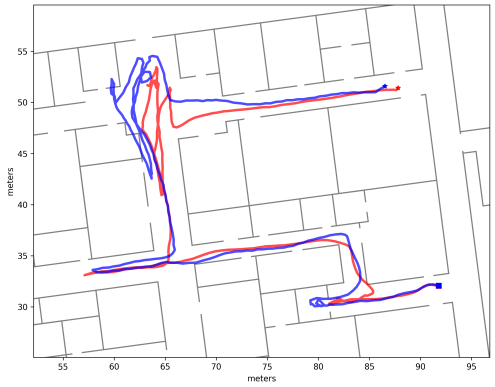
(c)

Figure 5.24: (a)–(b) Recorded route R1W using A/S (blue line) and RoNIN (red line) for (a): P1. (b): P2. (c) Synchronized screenshot of the app and P4's perspective, as he notices the staircases on his right side. The notification generated by the app is highlighted within an orange box. Experimenters reminded him to avoid ascending the stairs.

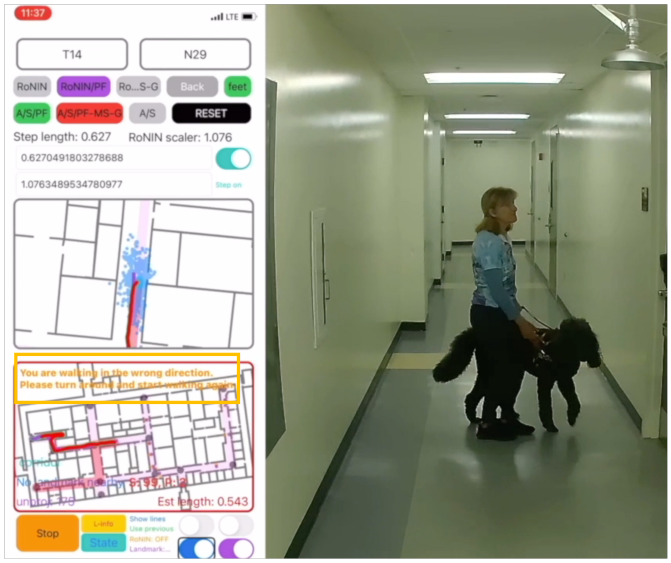
Certain cells in Table 5.6 are shaded in gray to indicate trials where participants

completed the trial successfully but occasionally had to retrace their path, as instructed by the WayFinding app, due to missed turns. Instances where users missed can be found in Figures 5.23 (b–c), 5.24 (b), 5.26, and 5.25 (b). Subsequent analysis, using both recorded video footage and app data, revealed the underlying causes for these missed turns. Regarding P7, who missed turns across multiple routes, it's worth mentioning that she was walking quite fast with her dog. By the time the notification concluded and she processed it, she had already passed the junction. Also, while navigating route R3W, P7 mistakenly entered a small open room shortly after starting the trial (as shown in Figure 5.25 (a)). Afterward, she missed several turns within the same route, even making the same mistake multiple times, before eventually finding the correct final route segment and reaching her destination (Figure 5.25 (b) and (c)). Some missing turn instances were attributed to external distractions, like interaction with passersby (P5 in R1W), or the participants being engaged in conversation when the notification was delivered (P2 in R2W; Figure 5.26 (a)). P6 missed a turn while traversing an open space in R3W(Figure 5.26 (b)). Potentially due to the absence of a nearby wall, she may have been uncertain about the precise turning point and continued until she sensed a wall to the right. By then, the app prompted her to turn around and get back on the correct route. On a single occasion (P6 in R2W), the app issued a notification just after the participant had gone past the junction, resulting in her missing the turn. This occurred because the localization algorithm briefly underestimated her position. Nonetheless, upon receiving the subsequent notification to turn around, she adeptly managed to navigate the missed turn.

(a)

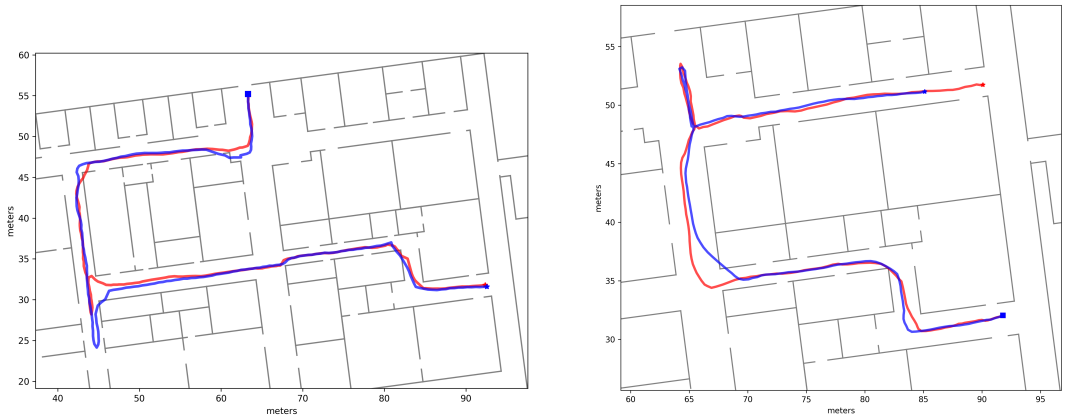


(b)



(c)

Figure 5.25: (a): P7 entering the room by mistake short after starting the route R3w. (b): Recorded R3W path traversed by p7 using A/S (blue line) and RoNIN (red line). (c): Synchronized screenshot of the app and P7's perspective, as she is turning around after hearing the "SW notification" that she is walking in the wrong direction. The notification generated by the app is highlighted within an orange box.



(a) (b)

Figure 5.26: Recorded paths using A/S (blue line) and RoNIN (red line). (a): P2, R2W. (b): P6, R3W.

Participants who used long canes demonstrated a proactive response to early notifications about upcoming turns. They tended to approach the wall on the side where the turn was expected, using their canes to tap the wall until they found an opening (as shown in Figure 5.27 (a)). In certain instances, there was an alcove (a recessed area in the wall) positioned just before or after the corridor junction. This caused a few participants to temporarily pause in the alcove before eventually finding their way out and continuing along their path (Figure 5.27 (b). An interesting deviation from this pattern was observed in the case of P2. Unlike the others, P2, an experienced independent traveler, employed a distinct mobility technique. Instead of scanning the ground in front of him, P2 used his cane to periodically tap the floor surface. He relied on the resulting sounds to make judgments about nearby surfaces. When alerted to an upcoming junction, he would

move closer to the corresponding side of the corridor (without tapping the wall) until he detected the presence of an opening to his left or right. At that point, he would execute a 90° turn into the opening. However, while he was traversing the R3W route, he missed a turn and had to retrace his steps following the system's guidance. He later commented, "I knew where that was exactly", suggesting he might have recognized the opening during his initial attempt.

(a) (b)

Figure 5.27: (a) P6 walks along the wall, searching for an opening to turn in route R1W. (b) In route R3W, P5 encountered an alcove on the right side after the last turn but managed to find his way.

The two participants with guide dogs exhibited distinct behaviors compared to those using canes, and each subject displayed unique characteristics. In theory, users can verbally communicate instructions like 'left' or 'right' from the app to their dogs, as dogs

are usually trained to comprehend these directions. However, P1 and P7 each adopted different approaches.

In the case of P1, she walked slowly, and the dog was guiding her cautiously. For example, in route R1W, she told the dog to turn "right" for the second turn, and since there was an alcove with the room's door in it, the dog decided to guide her into the room (Figure 5.28 (a)). The experimenter had to intervene to help her find her position, as marked "*E*" in Table 5.6. It is possible that if P1 had been using a cane, she could have explored to the left and right to find a way out of the alcove. The other issue was, upon receiving instructions through the app, she frequently needed to repeat the turn commands to her dog until it complied, and at times, she had to guide the dog gently. For instance, in another trial, R2W, the experimenter had to provide verbal assistance. At the start of R2W, they missed the initial turn because, by the time she commanded her dog to turn "right" after the notification, they had already passed the intersection. They correctly retraced their steps as advised by the app. However, when she instructed a "left" turn to return to the route, the dog declined and proceeded straight towards the exit door. To prevent them from leaving the building, we intervened. It took some time before P1 persuaded her dog to walk along that corridor again (Figure 5.28 (b)). After the study concluded, P1 mentioned that her dog might have felt nervous due to the trial circumstances.

As mentioned earlier, P7, on the other hand, was walking at a fast pace with her dog. This speed might have led her to pass by a junction without making the intended turn until she had heard the app's notification in its entirety, to then receive the "turn

around" notification (Figure 5.25 (c)).

(a)

(b)

Figure 5.28: (a): P1's Dog guided her to the alcove after the second turn in R1W; the experimenter helped her when she felt stuck there. (b): P1 repeated the turn right command to her dog while it was reluctant to turn.

Both A/S and RoNIN algorithms successfully tracked participants along most of the routes, except for the initial segment of R1W involving the first three participants. This occurred before adding the adaptive step length mechanism, as mentioned earlier. Occasionally, the reconstructed path showed some slight deviations, such as cutting corners (Figure 5.23 (a), RoNIN), or momentary overshooting of the location (Figure 5.24 (b), A/S). However, our Particle Filtering implementation's drift tracking mechanism appeared to function well in these trials, mitigating such issues. Notably, both algorithms successfully traced the complex path taken by P7 in R3W (Figure 5.25 (b)).

5.5.4 Questionnaire and Feedback

Table 5.7 presents the participants' responses to the System Usability Scale questionnaire, which was administered upon the completion of the trials. The questionnaire consists of 10 questions, each offering five response options ranging from "Strongly agree" (5) to "Strongly disagree" (1). The total score, following the SUS methodology proposed by [11], was recorded as 80.36. While there is an ongoing debate surrounding the interpretation of SUS scores [12], this score aligns with a percentile rank of 90% when compared to the score distribution documented in [88]. It's important to note that participants provided responses to the SUS questions and open-ended questions considering both the WayFinding and BackTracking apps, rather than answering each question separately for each individual app.

Question \ User	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 5.7: System Usability Scale (SUS) responses. The overall SUS score [11] was 80.36.

Below is a list of open-ended questions, along with a summary of the responses:

- **Do you think that the system always knew your location?** The majority of participants responded with variations of "yes, most of the time," with P7

indicating 99% accuracy and P1 reporting 80% accuracy. However, P2 and P3 answered "No". P2 mentioned occasional incorrect localization, while P3 felt that localization accuracy depended on walking style.

- **Do you think that the system gave you the correct directions?** P4 through P7 responded with an affirmative ”yes”. P2 believed the system provided accurate directions most of the time, similar to P1, who responded around 80% of the time. P3’s response was, ”Yes when it knew what it was doing”.

- **The system often gives turning directions (such as “at the next 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?** All participants indicated that this was generally not an issue, with some offering specific feedback. P1 mentioned, ”Once I got used to it, I sort of got it”. P2 remarked that it wasn’t a problem unless there were obstacles he could potentially bump into when moving closer to the wall in preparation for the turn. It’s worth noting that P2 had a distinctive approach using the cane and chose to walk farther from the wall. P4 shared that it created a problem, only once while traversing R2B using the BackTracking app, when she made a right turn too soon. P6 expressed that, while not a significant problem in itself, it would have been preferable if the notification timing were more ”consistent”, meaning the prompts always occurred at the same distance from the junction. However, due to localization errors, this distance often varied. P7 noted that this was the only aspect of the system that

didn't meet her expectations in terms of accuracy.

- **Were the notifications understandable? Too many notifications? Too few?** All participants expressed that the notifications were either "fine" or "just right". P2 explained more, mentioning that the system delivered the exact type of information he required: it offered an approximate distance to the upcoming turn and provided a heads-up just before the turn. P1 expressed a desire to know if there were any doors to open. However, it's important to clarify that the routes in the study did not require participants to open doors. There was a single open door in the middle of one corridor that they should have passed during the trial. As previously mentioned, landmark notifications were intentionally disabled for P1, so she did not receive a notification about it. Regarding landmark notifications, P2 believed that the number of landmark notifications during the trials was appropriate. However, he noted that if he was particularly focused on the route, he might prefer fewer notifications. In contrast, both P4 and P6 had highly positive feedback regarding landmark notifications. For example, P6 cited specific announcements like "staircase on the left' and "benches on the right" as examples of useful information. Similarly, P4 showed great enthusiasm for the landmark notifications and believed they could be practical in real-life situations. She mentioned scenarios like feeling thirsty and wanting to know if there was a drinking fountain nearby. She also pointed out that landmark notifications could be especially helpful for first-time visitors, with the option to disable them once

they become more familiar with the environment.

- **Was it easy for you to use the Watch?** Every participant, with the exception of P2, expressed that they found it easy. P2 described it as an "adventure" but also mentioned that with a bit more practice, he would have become more proficient at using the Watch gestures.

- **What would you like to have in this app that is not already there?** P1 inquired about how to locate the starting point and ensure the correct direction when beginning a route. P3 suggested the ability to scroll through a route description one step at a time. In the current implementation, a left swipe provides a description of the remaining route but doesn't allow the user to pause and review each step. P4 wished to manually add landmarks as needed, such as when passing a location of interest. Both P2 and P5 said that they would like to have more contextual information about the space they are visiting. Additionally, P5 indicated an interest in knowing their current facing direction, as it would be particularly valuable when feeling disoriented, providing them with independence in such situations, and reducing the need for external assistance.

- **Did you notice any difference between the WayFinding system and the BackTracking system?** Overall, the participants noted a high level of consistency between the two apps. P5 and P7 even went as far as to suggest that if they hadn't been informed about their distinctiveness, they might not have been able to tell that they were different apps. On the other hand, P2 and P3

were somewhat disappointed by the inconsistency in unit usage between the two apps. It's important to note that, for the first three participants (P1 to P3), the BackTracking app initially provided distances only in units of steps due to an implementation mistake. This issue was subsequently corrected for the following users.

- **Do you think that using this app would make you feel safer or more confident when traveling alone in a new place? [Asked for each app separately.]** With the exception of P3, all participants responded with an enthusiastic "yes". P3, however, considered the potential use of the Backtracking app in scenarios like navigating a conference with multiple tables in a large hall, such as when going to the restroom and returning to the same table. P2 imagined employing the app for navigating unfamiliar locations, particularly in vast complexes such as medical buildings. He expressed a preference for having a menu that could help him refine his destination selection. Even if the app couldn't guide him to the precise destination, he saw value in it bringing him closer to his intended location. P4 found value in the WayFinding app even for familiar buildings, like a large office complex she recently started visiting. She shared an experience about navigating this building where, while pausing to plan her route, she was approached by several bystanders offering unwelcome assistance. An app like this could be invaluable in moments of uncertainty like that. She added it can also help reduce the cognitive load of tracking her position and navigating, allowing her to focus on other tasks,

which is particularly useful in noisy environments. P5 noted that the apps helped him build a mental map of the entire route, eliminating the need to memorize the entire path because the apps provided prompts as he moved along it.

5.6 Conclusions

Inertial-based localization is advantageous for our indoor navigation application targeting visually impaired individuals. It operates independently of external infrastructure and doesn't require users to hold the smartphone, which is crucial as they often have a hand occupied with a cane or dog leash. Previous inertial sensor-based wayfinding systems for visually impaired individuals utilized the "user as a sensor" method to counter localization errors caused by accumulated drift [25, 6]. While effective in uncertain situations [82], this approach places a significant burden on users who must detect landmarks. We explored the feasibility of a WayFinding system relying solely on inertial sensor data, without user input. Our findings suggest its viability, especially in buildings with interconnected corridors.

Inertial systems relying on dead-reckoning can introduce significant localization errors. To address this, the WayFinding app used particle filtering to maintain route consistency with the building layout. Our study environment featured a mix of narrow and wide corridors along with spacious open areas, resembling typical public spaces like office buildings, schools, and health centers. However, in larger complexes like airports or vast shopping malls, inertial-only navigation systems could face challenges.

Within expansive open spaces, particles disperse uniformly, limiting the Particle Filter's effectiveness in mitigating drift. In our study environment, corridor junctions were limited to 90° intersections.

We conducted a user study with 7 blind participants, including 5 cane users and 2 relying on guide dogs. Users found the Watch interface user-friendly, even if unfamiliar with SmartWatches, resulting in positive feedback. In real-life scenarios, users might need to remove their phones for tasks like calls or messages. A/S assumes fixed phone orientation, whereas RoNIN, without this constraint, suits natural phone interactions better. The study confirmed inertial-based wayfinding feasibility, with all users completing the routes. Qualitative trajectory analysis (as shown in Figures 5.23-5.25) did not indicate a clear winner, suggesting that both techniques can be effective with a well-implemented particle filter.

Our system offers advanced notifications to guide users when approaching junctions. One aim of our study was to assess the feasibility and acceptance of this mechanism. Previous research, such as the work in [2] on sonification techniques, has explored effective methods for ensuring that pedestrians initiate their turns at the correct moment. While these techniques are suitable for more accurate localization systems like BLE beaconing [78], our system's reliance on inertial sensing, with its inherent localization errors, necessitates early notifications. Although we verified the mechanism performance in pre-experiment tests, we were still uncertain whether blind participants would struggle to locate the exact junction or if they'd find this approach bothersome. However, we were pleased to observe that, overall, participants navigated these situations successfully, even

when facing structural obstacles, such as being stuck in an alcove while searching for an opening. Additionally, participants expressed in their responses to the 3rd open-ended question that advance notification wasn't a problem. The positive usability scores from the System Usability Scale (SUS) responses further indicate overall satisfaction with the interface design. However, it's crucial to acknowledge potential limitations, such as instances where two close openings exist on the same side of a corridor, which may lead to confusion. In such cases, the system can enhance user clarity by providing additional contextual information, like notifying the user about the presence of two junctions and recommending the second one.

The primary limitation of our approach is the requirement for users to initiate a route from a specific starting point and continue in a predetermined direction. This initial phase is essential to align the user's path with the floor plan frame. The identical requirement exists in prior research on inertial-based wayfinding [25, 6], arising from the inherent dead-reckoning characteristics of this method. Two potential solutions could address this challenge. One option involves utilizing the "user as a sensor" paradigm, where users begin their journey from an easily identifiable location, such as the main entrance of a building, or from a distinctive landmark, like the base of a staircase, and receive guidance to walk straight ahead from there. Creating a hybrid system that uses visual data, potentially through automated landmark recognition [17], to make occasional corrections using computer vision techniques when necessary for determining the user's location and orientation could be an alternative. After these adjustments, users can resume using the inertial system for tracking by keeping their phones in their pockets.

Finally, we were encouraged to learn from our participants that the tested apps produced feelings of increased safety and confidence in their independent travel. Although we acknowledge the existing system limitations, this feedback, combined with the positive SUS questionnaire scores, reinforces the significant potential of our proposed technology as a practical navigation aid.

Bibliography

[1] Drishti: an integrated indoor/outdoor blind navigation system and service. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*, pages 23–30. IEEE, 2004.

[2] Dragan Ahmetovic, Federico Avanzini, Adriano Baratè, Cristian Bernareggi, Marco Ciardullo, Gabriele Galimberti, Luca A Ludovico, Sergio Mascetti, and Giorgio Presti. Sonification of navigation instructions for people with visual impairment. *International Journal of Human-Computer Studies*, 177:103057, 2023.

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

[4] Dragan Ahmetovic, Masayuki Murata, Cole Gleason, Erin Brady, Hironobu Takagi, Kris Kitani, and Chieko Asakawa. Achieving practical and accurate indoor navigation for people with visual impairments. In *Proceedings of the 14th International Web for All Conference*, pages 1–10, 2017.

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

[6] Ilias Apostolopoulos, Navid Fallah, Eelke Folmer, and Kostas E Bekris. Integrated online localization and navigation for people with visual impairments using smart phones. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(4):1–28, 2014.

[7] Shahid Ayub, Xiaowei Zhou, Soroush Honary, Alireza Bahraminasab, and Bahram Honary. Indoor pedestrian displacement estimation using smart phone inertial sensors. *International Journal of Innovative Computing and Applications*, 4(1):35–42, 2012.

[8] Stéphane Beauregard and Harald Haas. Pedestrian dead reckoning : A basis for personal positioning. 2006.

[9] Primož Bencak, Darko Hercog, and Tone Lerher. Indoor Positioning System Based on Bluetooth Low Energy Technology and a Nature-Inspired Optimization Algorithm. *Electronics*, 11(3), 2022.

[10] Johann Borenstein, Lauro Ojeda, and Surat Kwanmuang. Heuristic reduction of gyro drift for personnel tracking systems. *The Journal of navigation*, 62(1):41–58, 2009.

[11] John Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3):189–194, 1996.

[12] John Brooke. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013.

[13] Rastislav Cervenak and Pavel Masek. Arkit as indoor positioning system. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–5. IEEE, 2019.

[14] Changhao Chen, Chris Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. IONet: Learning to Cure the Curse of Drift in Inertial Odometry. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[15] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, A. Markham, and Agathoniki Trigoni. OxIOD: The Dataset for Deep Inertial Odometry. *ArXiv*, abs/1809.07491, 2018.

[16] Hao Chen, Yifan Zhang, Wei Li, Xiaofeng Tao, and Ping Zhang. Confi: Convolutional neural networks based indoor Wi-Fi localization using channel state information. *Ieee Access*, 5:18066–18074, 2017.

[17] Lidong Chen, Yin Zou, Yaohua Chang, Jinyun Liu, Benjamin Lin, and Zhigang 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)*, pages 311–316. IEEE, 2019.

[18] Ruizhi Chen and Liang Chen. Smartphone-Based Indoor Positioning Technologies. In *Urban Informatics*, pages 467–490. Springer, 2021.

[19] Ruizhi Chen, Ling Pei, and Yuwei Chen. A smart phone based PDR solution for indoor navigation. In *Proceedings of the 24th international technical meeting of the satellite division of the institute of navigation (ION GNSS 2011)*, pages 1404–1408, 2011.

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

[21] Seyed Ali Cheraghi, Vinod Namboodiri, and Laura Walker. Guidebeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 121–130. IEEE, 2017.

[22] Sakmongkon Chumkamon, Peranitti Tuvaphanthaphiphat, and Phongsak Keeratiwintakorn. A blind navigation system using rfid for indoor environments. In *2008 5th international conference on electrical engineering/electronics, computer, telecommunications and information technology*, volume 2, pages 765–768. IEEE, 2008.

[23] Marcus Edel and Enrico Köppe. An advanced method for pedestrian dead reckoning

using BLSTM-RNNs. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6, 2015.

[24] Yuki Endo, Kei Sato, Akihiro Yamashita, and Katsushi Matsubayashi. Indoor positioning and obstacle detection for visually impaired navigation system based on lsd-slam. In *2017 International Conference on Biometrics and Kansei Engineering (ICBAKE)*, pages 158–162. IEEE, 2017.

[25] Navid Fallah, Ilias Apostolopoulos, Kostas Bekris, and Eelke Folmer. The user as a sensor: Navigating users with visual impairments in indoor spaces using tactile landmarks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 425–432, New York, NY, USA, 2012. Association for Computing Machinery.

[26] Ramsey Faragher and Robert Harle. An analysis of the accuracy of bluetooth low energy for indoor positioning applications. In *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, pages 201–210, 2014.

[27] José Faria, Sérgio Lopes, Hugo Fernandes, Paulo Martins, and João Barroso. Electronic white cane for blind people navigation assistance. In *2010 World Automation Congress*, pages 1–7. IEEE, 2010.

[28] Carl Fischer, Poorna Talkad Sukumar, and Mike Hazas. Tutorial: Implementing a

pedestrian tracker using inertial sensors. *IEEE pervasive computing*, 12(2):17–27, 2012.

[29] German Flores and Roberto Manduchi. Easy return: An app for indoor backtracking assistance. In *ACM CHI Conference on Human Factors in Computing Systems (CHI'18)*, 04/2018 2018.

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

[31] Germán H. Flores, Roberto Manduchi, and Enrique D. Zenteno. Ariadne's thread: Robust turn detection for path back-tracing using the iphone. In *2014 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, pages 133–140, 2014.

[32] V Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, 2(3):24–33, 2003.

[33] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications*, 25(6):38–46, 2005.

[34] Giovanni Fusco and James M Coughlan. Indoor localization using computer vision and visual-inertial odometry. In *Computers Helping People with Special Needs: 16th International Conference, ICCHP 2018, Linz, Austria, July 11-13, 2018, Proceedings, Part II 16*, pages 86–93. Springer, 2018.

[35] Giovanni Fusco and James M Coughlan. Indoor localization for visually impaired travelers using computer vision on a smartphone. In *Proceedings of the 17th International Web for All Conference*, pages 1–11, 2020.

[36] Enrique García, Pablo Poudereux, Álvaro Hernández, Jesús Ureña, and David Gualda. A robust UWB indoor positioning system for highly complex environments. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 3386–3391. IEEE, 2015.

[37] Davoud Gholamiangonabadi, Nikita Kiselov, and Katarina Grolinger. Deep neural networks for human activity recognition with wearable sensors: Leave-one-subject-out cross-validation for model selection. *Ieee Access*, 8:133982–133994, 2020.

[38] Stéphanie Giraud, Anke M Brock, Marc J-M Macé, and Christophe Jouffrais. Map learning with a 3d printed interactive small-scale model: Improvement of space and text memorization in visually impaired students. *Frontiers in psychology*, 8:930, 2017.

[39] Fuqiang Gu, Kourosh Khoshelham, Jianga Shang, Fangwen Yu, and Zhuo Wei. Robust and accurate smartphone-based step counting for indoor localization. *IEEE Sensors Journal*, 17(11):3453–3460, 2017.

[40] Fuqiang Gu, Kourosh Khoshelham, Chunyang Yu, and Jianga Shang. Accurate step length estimation for pedestrian dead reckoning localization using stacked

autoencoders. *IEEE Transactions on Instrumentation and Measurement*, 68(8):2705–2713, 2019.

[41] Yanying Gu, Anthony Lo, and Ignas Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys Tutorials*, 11(1):13–32, 2009.

[42] João Guerreiro, Daisuke Sato, Dragan Ahmetovic, Eshed Ohn-Bar, Kris M Kitani, and Chieko Asakawa. Virtual navigation for blind people: Transferring route knowledge to the real-world. *International Journal of Human-Computer Studies*, 135:102369, 2020.

[43] Tiago Guerreiro, Kyle Montague, João Guerreiro, Rafael Nunes, Hugo Nicolau, and Daniel JV Gonçalves. Blind people interacting with large touch surfaces: Strategies for one-handed and two-handed exploration. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, pages 25–34, 2015.

[44] Ann Hallemans, Els Ortibus, Francoise Meire, and Peter Aerts. Low vision affects dynamic stability of gait. *Gait & posture*, 32(4):547–551, 2010.

[45] Sachini Herath, Hang Yan, and Yasutaka Furukawa. Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3146–3152. IEEE, 2020.

[46] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[48] Andreas Hub, Joachim Diepstraten, and Thomas Ertl. Design and development of an indoor navigation and object identification system for the blind. *ACM Sigaccess Accessibility and Computing*, (77-78):147–152, 2003.

[49] Marco Iosa, Augusto Fusco, Giovanni Morone, Stefano Paolucci, et al. Effects of visual deprivation on gait dynamic stability. *The Scientific World Journal*, 2012, 2012.

[50] Tatsuya Ishihara, Jayakorn Vongkulbhisal, Kris M Kitani, and Chieko Asakawa. Beacon-guided structure from motion for smartphone-based navigation. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 769–777. IEEE, 2017.

[51] Sampath Jayalath and Nimsiri Abhayasinghe. A gyroscopic data based pedometer algorithm. In *2013 8th International Conference on Computer Science Education*, pages 551–555, 2013.

[52] Antonio Ramón Jiménez, Fernando Seco, José Carlos Prieto, and Jorge Guevara.

Indoor pedestrian navigation using an INS/EKF framework for yaw drift reduction and a foot-mounted IMU. In *2010 7th workshop on positioning, navigation and communication*, pages 135–143. IEEE, 2010.

[53] Jiheon Kang, Joonbeom Lee, and Doo-Seop Eom. Smartphone-based traveled distance estimation using individual walking patterns for indoor localization. *Sensors*, 18(9):3149, 2018.

[54] Jeong Won Kim, Han Jin Jang, Dong-Hwan Hwang, and Chansik Park. A step, stride and heading determination for the pedestrian navigation system. *Journal of Global Positioning Systems*, 01:0–0, 2004.

[55] Itzik Klein and Omri Asraf. StepNet—Deep Learning Approaches for Step Length Estimation. *IEEE Access*, 8:85706–85713, 2020.

[56] Seung-uk Ko, Magdalena I Tolea, Jeffrey M Hausdorff, and Luigi Ferrucci. Sex-specific differences in gait patterns of healthy older adults: results from the baltimore longitudinal study of aging. *Journal of biomechanics*, 44(10):1974–1979, 2011.

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

[58] Pavel Kriz, Filip Maly, and Tomas Kozel. Improving indoor localization using bluetooth low energy beacons. *Mobile information systems*, 2016, 2016.

[59] Andreas Kunz, Klaus Miesenberger, Limin Zeng, and Gerhard Weber. Virtual

navigation environment for blind and low vision people. In *International Conference on Computers Helping People with Special Needs*, pages 114–122. Springer, 2018.

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

[61] Quentin Ladetto. On foot navigation: Continuous step calibration using both complementary recursive prediction and adaptive kalman filtering, 2000.

[62] Orly Lahav, Hadas Gedalevitz, Steven Battersby, David Brown, Lindsay Evett, and Patrick Merritt. Virtual environment navigation with look-around mode to explore new real spaces by people who are blind. *Disability and rehabilitation*, 40(9):1072–1084, 2018.

[63] Douglas J Leith and Stephen Farrell. Measurement-based evaluation of Google/Apple exposure notification API for proximity detection in a commuter bus. *Plos one*, 16(4):e0250826, 2021.

[64] Fabrizio Leo, Elena Cocchi, and Luca Brayda. The effect of programmable tactile displays on spatial learning skills in children and adolescents of different visual disability. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(7):861–872, 2016.

[65] Bing Li, J Pablo Munoz, Xuejian Rong, Jizhong Xiao, Yingli Tian, and Aries Arditi. Isana: wearable context-aware indoor assistive navigation with obstacle avoidance for the blind. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, pages 448–462. Springer, 2016.

[66] Wenxin Liu, David Caruso, Eddy Ilg, Jing Dong, Anastasios I. Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel. TLIO: Tight Learned Inertial Odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, Oct 2020.

[67] Dierna Giovanni Luca and Macha Alberto. Towards accurate indoor localization using ibeacons, fingerprinting and particle filtering. In *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016.

[68] Luca Mainetti, Luigi Patrono, and Ilaria Sergi. A survey on indoor positioning systems. In *2014 22nd international conference on software, telecommunications and computer networks (SoftCOM)*, pages 111–120. IEEE, 2014.

[69] Roberto Manduchi and Sri Kurniawan. *Assistive technology for blindness and low vision*. CRC Press, 2018.

[70] Roberto Manduchi, Sri Kurniawan, and Homayoun Bagherinia. Blind guidance using mobile computer vision: A usability study. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*, pages 241–242, 2010.

[71] Sergio Mascetti, Dragan Ahmetovic, Andrea Gerino, and Cristian Bernareggi. Zebrarecognizer: Pedestrian crossing recognition for people with visual impairment or blindness. *Pattern Recognition*, 60:405–419, 2016.

[72] Rainer Mautz. Overview of current indoor positioning systems. *Geodezija ir kartografija*, 35(1):18–22, 2009.

[73] Emiliano Miluzzo, Michela Papandrea, Nicholas D Lane, Hong Lu, and Andrew T Campbell. Pocket, bag, hand, etc.-Automatically detecting phone context through discovery. *Proc. PhoneSense 2010*, pages 21–25, 2010.

[74] Fatemeh Mirzaei and Roberto Manduchi. In–vehicle positioning for public transit using ble beacons. *Indoor Positioning and Indoor Navigation (IPIN 21)-Work-In-Progress papers*, 2021.

[75] Fatemeh Mirzaei and Roberto Manduchi. Interpersonal proximity detection using RSSI–based techniques. *Indoor Positioning and Indoor Navigation (IPIN 21)-Work-In-Progress papers*, 2021.

[76] Masayuki Murata, Dragan Ahmetovic, Daisuke Sato, Hironobu Takagi, Kris M Kitani, and Chieko Asakawa. Smartphone-based indoor localization for blind navigation across building complexes. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. IEEE, 2018.

[77] Vishnu Nair, Greg Olmschenk, William H Seiple, and Zhigang Zhu. Assist: Evalu-

ating the usability and performance of an indoor navigation assistant for blind and visually impaired people. *Assistive Technology*, 34(3):289–299, 2022.

[78] Eshed Ohn-Bar, João Guerreiro, Kris Kitani, and Chieko Asakawa. Variability in reactions to instructional guidance during smartphone-based assisted navigation of blind users. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies*, 2(3):1–25, 2018.

[79] Venet Osmani, Iacopo Carreras, Aleksandar Matic, and Piret Saar. An analysis of distance estimation to detect proximity in social interactions. *Journal of Ambient Intelligence and Humanized Computing*, 5:297–306, 2014.

[80] Romedi Passini and Guyltne Proulx. Wayfinding without vision: An experiment with congenitally totally blind people. *Environment and behavior*, 20(2):227–252, 1988.

[81] Ren Peng, Fatemeh Elyasi, and R Manduchi. Smartphone-Based Inertial Odometry for Blind Walkers. *Sensors*, 21, 06/2021 2021.

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

[83] Valérie Renaudin, Melania Susi, and Gérard Lachapelle. Step length estimation using handheld inertial sensors. *Sensors*, 12(7):8507–8525, 2012.

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

[85] Timothy H Riehle, P Lichter, and Nicholas A Giudice. An indoor navigation system to support the visually impaired. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4435–4438. IEEE, 2008.

[86] Daisuke Sato, Uran Oh, João Guerreiro, Dragan Ahmetovic, Kakuya Naito, Hironobu Takagi, Kris M Kitani, and Chieko Asakawa. Navcog3 in the wild: Large-scale blind indoor navigation assistant with semantic features. *ACM Transactions on Accessible Computing (TACCESS)*, 12(3):1–30, 2019.

[87] Daisuke Sato, Uran Oh, Kakuya Naito, Hironobu Takagi, Kris Kitani, and Chieko 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*, pages 270–279, 2017.

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

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

[90] Frederik Schrooyen, Isabel Baert, Steven Truijen, Luc Pieters, Tim Denis, Koen Williame, and Maarten Weyn. Real time location system over wifi in a healthcare environment. *Journal on Information Technology in Healthcare*, 4(6):401–416, 2006.

[91] Noboru Sekiya, Hiroshi Nagasaki, Hajime Ito, and Taketo Furuna. Optimal walking in terms of variability in step length. *Journal of Orthopaedic & Sports Physical Therapy*, 26(5):266–272, 1997.

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

[93] Farzaneh Shahini, Vanessa Nasr, and Maryam Zahabi. A friendly indoor navigation app for people with disabilities (find). In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 66, pages 1922–1926. SAGE Publications Sage CA: Los Angeles, CA, 2022.

[94] Arno Solin, Santiago Cortes, Esa Rahtu, and Juho Kannala. Inertial odometry on handheld smartphones. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1–5. IEEE, 2018.

[95] Rakesh Soni and Samir Trapasiya. A survey of step length estimation models

based on inertial sensors for indoor navigation systems. *International Journal of Communication Systems*, page e5053, 2021.

[96] Sebastijan Sprager and Matjaz B Juric. Inertial sensor-based gait recognition: A review. *Sensors*, 15(9):22089–22127, 2015.

[97] William Storms, Jeremiah Shockley, and John Raquet. Magnetic field navigation in an indoor environment. In *2010 Ubiquitous Positioning Indoor Navigation and Location Based Service*, pages 1–10. IEEE, 2010.

[98] Poorna Talkad Sukumar. Enhanced Stance Phase Detection and Extended Kalman Filtering for Strapdown Pedestrian Dead Reckoning MSc in Mobile and Ubiquitous Computing. 2010.

[99] Scott Sun, Dennis Melamed, and Kris Kitani. IDOL: Inertial deep orientation-estimation and localization. *arXiv preprint arXiv:2102.04024*, 2021.

[100] Yi Sun, Huaming Wu, and Jochen Schiller. A step length estimation model for position tracking. In *2015 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6. IEEE, 2015.

[101] Brandon Taylor, Anind Dey, Dan Siewiorek, and Asim Smailagic. Customizable 3d printed tactile maps as interactive overlays. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 71–79, 2016.

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

[103] Mônica SV Tomomitsu, Angelica Castilho Alonso, Eurica Morimoto, Tatiana G Bobbio, and Julia Greve. Static and dynamic postural control in low-vision and normal-vision adults. *Clinics*, 68:517–521, 2013.

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

[105] Chia Hsuan Tsai, Peng Ren, Fatemeh Elyasi, and Roberto 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)*, pages 117–122. IEEE, 2021.

[106] Simon Ungar. Cognitive mapping without visual experience. In *Cognitive Mapping*, pages 221–248. Routledge, 2018.

[107] Frank Van Diggelen and Per Enge. The world's first gps mooc and worldwide laboratory using smartphones. In *Proceedings of the 28th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2015)*, pages 361–369, 2015.

[108] Tom Van Haute, Eli De Poorter, Pieter Crombez, Filip Lemic, Vlado Handziski, Niklas Wirström, Adam Wolisz, Thiemo Voigt, and Ingrid Moerman. Performance analysis of multiple indoor positioning systems in a healthcare environment. *International journal of health geographics*, 15(1):1–15, 2016.

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

[110] Qu Wang, Langlang Ye, Haiyong Luo, Aidong Men, Fang Zhao, and Yan Huang. Pedestrian stride-length estimation based on lstm and denoising autoencoders. *Sensors*, 19(4):840, 2019.

[111] Harvey Weinberg. Using the adxl 202 in pedometer and personal navigation applications. 2002.

[112] William R Wiener, Richard L Welsh, and Bruce B Blasch. *Foundations of orientation and mobility*, volume 1. American Foundation for the Blind, 2010.

[113] Michele A Williams, Caroline Galbraith, Shaun K Kane, and Amy Hurst. " just let the cane hit it" how the blind and sighted see navigation differently. In *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility*, pages 217–224, 2014.

[114] Michele A Williams, Amy Hurst, and Shaun K Kane. " pray before you step out" describing personal and situational blind navigation behaviors. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–8, 2013.

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

[116] Dayu Yan, Chuang Shi, and Tuan Li. An improved PDR system with accurate heading and step length estimation using handheld smartphone. *Journal of Navigation*, page 1–19, 2021.

[117] Hang Yan, Qi Shan, and Yasutaka Furukawa. RIDI: Robust IMU Double Integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[118] Yingbiao Yao, Lei Pan, Wei Fen, Xiaorong Xu, Xuesong Liang, and Xin Xu. A robust step detection and stride length estimation for pedestrian dead reckoning using a smartphone. *IEEE Sensors Journal*, 20(17):9685–9697, 2020.

[119] Takuto Yoshida, Junto Nozaki, Kenta Urano, Kei Hiroi, Katsuhiko Kaji, Takuro Yonezawa, and Nobuo Kawaguchi. Sampling rate dependency in pedestrian walking speed estimation using DualCNN-LSTM. In Robert Harle, Katayoun Farrahi, and Nicholas D. Lane, editors, *Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, UbiComp/ISWC 2019 Adjunct, London, UK, September 9-13, 2019*, pages 862–868. ACM, 2019.

[120] Tuo Yu, Haiming Jin, and Klara Nahrstedt. Shoesloc: In-shoe force sensor-based indoor walking path tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(1):1–23, 2019.

[121] Long Zhao, Zhen Liu, Tiejun Li, Baoqi Huang, and Lihua Xie. Moving target

positioning based on a distributed camera network. *Mathematical Problems in Engineering*, 2014, 2014.

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