sensors

MDPI

Article

Smartphone-Based Inertial Odometry for Blind Walkers

Peng Ren *, Fatemeh Elyasi and Roberto Manduchi

Computer Science and Engineering, UC Santa Cruz, Santa Cruz, CA 95064, USA; felyasi@ucsc.edu (F.E.); manduchi@soe.ucsc.edu (R.M.)
* Correspondence: pren1@ucsc.edu

Abstract: Pedestrian tracking systems implemented in regular smartphones may provide a convenient mechanism for wayfinding and backtracking for people who are blind. However, virtually all existing studies only considered sighted participants, whose gait pattern may be different from that of blind walkers using a long cane or a dog guide. In this contribution, we present a comparative assessment of several algorithms using inertial sensors for pedestrian tracking, as applied to data from WeAllWalk, the only published inertial sensor dataset collected indoors from blind walkers. We consider two situations of interest. In the first situation, a map of the building is not available, in which case we assume that users walk in a network of corridors intersecting at 45° or 90°. We propose a new two-stage turn detector that, combined with an LSTM-based step counter, can robustly reconstruct the path traversed. We compare this with RoNIN, a state-of-the-art algorithm based on deep learning. In the second situation, a map is available, which provides a strong prior on the possible trajectories. For these situations, we experiment with particle filtering, with an additional clustering stage based on mean shift. Our results highlight the importance of training and testing inertial odometry systems for assisted navigation with data from blind walkers.

Keywords: inertial odometry; wayfinding; indoor pedestrian tracking

Citation: Ren, P.; Elyasi, F.; Manduchi, R. Smartphone-Based Inertial Odometry for Blind Walkers. *Sensors* **2021**, *21*, 4033. https://doi.org/10.3390/s21124033

Academic Editors: Joaquín Torres-Sospedra, Antoni Perez-Navarro and Raúl Montoliu

Received: 3 May 2021
Accepted: 9 June 2021
Published: 11 June 2021

1. Introduction

Smartphone-based odometry systems for pedestrian tracking in indoor, GPS-denied environments have received considerable attention in recent years. These systems may help a person reach a gate in an airport [1] or a shop in a mall [2], navigate a museum [3], or find one's car in a parking lot [4]. Among the various approaches considered in the literature, technology based on inertial sensors have a number of practical advantages. For example, inertial-based odometry does not require the installation of infrastructure such as Bluetooth low energy (BLE) beacons [5]. In addition, no prior calibration ("fingerprinting") is necessary, unlike for systems based on Wi-Fi [6] or BLE beacons. Compared with systems that use a camera to determine the user's location (visual-based odometry [7]), and that, thus, require good unoccluded visibility of the scene, inertial systems are able to track the user even when they keep the phone in their pocket. The downside of this modality is that the user's location is tracked by integrating inertial data, which leads to possibly large errors due to accumulated drift. A number of strategies to deal with drift have been proposed, including zero-velocity updates [8], spatial constraints (e.g., Bayes filtering using a map of the environment [9]), and machine learning [10]. Multiple well-calibrated inertial datasets (containing data from accelerometer and gyros) collected from regular smartphones carried by human walkers have been made available in recent years [10–12].

Among the communities of potential users of this technology, blind travelers arguably stand to benefit the most. Wayfinding can be extremely challenging for those without sight, who do not have access to landmarks and other visual information. Blind individuals can easily get lost in unfamiliar environments, which may discourage independent travel for these individuals. Reliable pedestrian tracking systems could improve safety and confidence for blind people navigating a shopping mall, a hospital, or a transit hub.

Although, in principle, the same mechanisms designed for sighted people could also be used by blind travelers, it is well known that the gait characteristics of blind individuals using a long (white) cane or a dog guide are different from those of sighted walkers [13–16]. For example, use of a long cane (e.g., using the two-point touch technique [17]) may result in large side-to-side swings. When "shorelining" a wall to maintain a straight trajectory, vibrations from the cane hitting the wall may be picked up by the smartphone's accelerometer. Walking without sight often leads to bumping onto obstacles or even onto people, requiring one to stop and re-orient themselves. These events (combined with other situations, such as negotiating a doorway or opening a door) contribute spurious inertial measurements that could challenge odometry algorithms designed for "cleaner" data associated with sight-assisted walking. Hence, it is important to evaluate these systems on "real" data from blind walkers. In this contribution, we use the WeAllWalk dataset [16], which is the only publicly available collection of inertial data from blind individuals. WeAllWalk contains inertial data from 10 blind participants, each carrying two iPhones, who traversed a number of paths in two different buildings using a long cane or a dog guide as travel aid.

We consider two realistic situations in this work. In one situation (*map-less*), no map of the environment is available. Even without a map, pedestrian tracking could be useful for *assisted return*: In this case, the system can provide spatial information and direction to a blind user who is backtracking their path. [18–21]. In the second situation (*map-assisted*), we assume that a map of the place is available in digital form. Clearly, this is the most desirable case, as it enables the system to provide turn-by-turn directions to the desired destination. In addition, a map provides a strong constraint on the reconstructed trajectory (e.g., the trajectory cannot go through a wall). In both cases, we consider two different types of pedestrian dead-reckoning (PDR) systems: a simple PDR using an RNN-based step counter, coupled with (possibly drift-corrupted) heading information as provided by the iPhone's sensors; and a more sophisticated system (RoNIN [10]), based on deep learning, that was shown to produce state-of-the-art results when tested with existing inertial datasets. Since RoNIN was trained with sighted walkers, we experimented with fine-tuning its network with data from blind walkers in WeAllWalk. For the *map-less* case, in which the strong constraint of wall impassibility cannot be relied on, we experimented with a simple turn/segment path representation. This is appropriate for buildings with corridors that intersect at discrete turning angles at multiples of 45° or 90°. Besides providing a strong spatial prior which can be used to compensate for drift, this representation is particularly useful for the verbal description of a path [18]. As an example, a path could be represented as "walk straight for 80 steps, then make a right turn, walk straight for 50 more steps, make a left turn, then after 40 steps you will reach your destination". It is important to note that for a turn/segment representation to be successful, turns must be detected robustly, which may be challenging in some situations. For example, a blind walker may stop and turn around to get their bearings, or to listen to a sound that may help with orientation, something that could mistakenly be interpreted by the system as a path turn [18]. Blind walkers (especially those who do not use a dog guide) often tend to veer when attempting to walk on a straight line, and this unwanted veering may generate false turn detections. We introduce a two-stage turn detection system, formed by an orientation tracker and a straight walking detector, that proved reliable in the face of such adversarial situations. For the *map-assisted* case, we integrate the two considered systems with a standard particle filter, which was enhanced with a module that finds the mode of the posterior state distribution.

These are the main contributions of this work:

- We introduce a new mechanism (based on a recurrent neural network) to robustly identify when a user is walking regularly along a straight path;
- We propose a new algorithm for turn detection that is unaffected by drift. This algorithm finds the difference in user's orientation between the beginning of a "straight

walking" segment, and the end of the prior segment. Orientation drift is tracked by a Mixture Kalman Filter;
- We describe a new method to identify the modes of the posterior probability distribution of the user's location tracked by a Particle Filter;
- We study the performance of different mechanisms of path reconstruction (map-less and map-assisted), as well as of an LSTM-based step counter of our turn detector, when applied to the data collected from the blind walkers represented in the WeAllWalk dataset. This analysis was conducted using a variety of assessment metrics. Our results highlight the importance of training these algorithms with data from the same community of users (in this case, blind walkers) these systems are designed for.

This article is organized as follows. We first review the related work (including the WeAllWalk dataset) in Section 2. The step counting, turn detection, and path reconstruction algorithms used in this work are presented in Section 3. Experiments are described in Section 4, where we also introduce the considered training and test modalities. These experimental results are then discussed in Section 5. Section 6 has the conclusions.

2. Related Works

2.1. Pedestrian Dead Reckoning (PDR)

Perhaps the simplest method to track the location of a walker is to count steps while measuring the user's orientation at all times [22–28]. Step counting is traditionally performed by finding peaks or other features in acceleration or rotation rate signals (e.g., [29–31]). More recently, recurrent neural networks (RNN) have been proposed as a robust alternative to "hand-crafted" algorithms [32–34].

The orientation of the phone can be obtained by proper integration of the data from the accelerometer and gyro [35,36], but this typically results in accumulated drift. Turns can be detected, for example, by measuring short-time variations of the azimuth angle (or of the rotation rate from the gyro [37]), which are unaffected by slowly varying drift. Flores et al. [38] proposed a system based on dynamic programming to estimate the discrete walker's orientation along with drift. Although effective in the tests of [38] with sighted walkers, this algorithm gave poor results with blind walkers [16]. Another problem is how to decouple the orientation of the phone from the direction of walking. A number of algorithms for the estimation of the direction of walking, independently of the orientation of the phone, have been developed [39–42]. Another topic of interest is the robust detection of steps and of stride lengths, which are used as a proxy for the walker's velocity [29–31], [43–48]. When a map of the environment is available, it may provide a strong constraint for the space of possible trajectories. Bayes filtering (in particular, particle filtering [9]) is normally used in these situations [49–52].

2.2. Learning-Based Odometry

In recent years, a number of data-driven techniques for odometry, that rely less on models and more on machine learning, have emerged. For example, RIDI [11] regresses user velocity from the time series of linear accelerations and angular velocities. IONet [53] uses a deep neural network to compute user velocity and heading. RoNIN [10] processes inertial data in a heading-agnostic reference frame using a variety of deep network architectures. Importantly, RoNIN is able to decouple the phone's orientation from the user's orientation when walking. This means that tracking is unaffected by any possible repositioning of the phone (e.g., if the user moves the phone to a different pocket). A number of other learning-based algorithms for computing the walker's velocity, or for detecting steps and measuring stride lengths, have been recently proposed [33,34,54–61].

2.3. Inertial Navigation for Blind Travelers

Wayfinding systems for blind individuals have received special attention, given the potential of this technology to enable independent travel. Although GPS can be used for localization in the outdoors, indoor environments (office buildings, shopping malls, transit

hubs) can be particularly daunting for blind travelers. By tracking the location of the blind walker, a wayfinding system can provide turn-by-turn directions, present location-based information (e.g., announce the presence of a store nearby), or help users re-trace their path from the starting point. Several wayfinding techniques specifically designed for blind travelers [62] have been considered, including visual odometry (e.g., [63,64]), BLE beacon-assisted navigation [65], and magnetic navigation [66]. Although inertial data are often used to integrate information from other sensors, relatively little research work has explored the use of inertial sensors alone for blind navigation. This includes work by Fallah et al. [67] and Riehle et al. [68]. Flores and Manduchi [18] proposed an inertial system for assisted return (backtracking) that used a turns or steps representation of indoor environments in a study with six blind participants.

2.4. Inertial Data Sets

Well-calibrated datasets are important for performance assessment of odometry algorithms, and are essential for the training of learning-based systems. RIDI [11] was built using a smartphone equipped with an RGBD camera, which enables complete pose (location+orientation) reconstruction at each time using Visual SLAM. OxIOD [12] is a data collection with four different phone placements. The walkers' location was tracked by a Vicon system. RoNIN [10] is a large dataset with inertial data collected from a smartphone. A different phone, equipped with an RGBD camera, was used for ground-truth walker pose estimation. More limited in scope, the dataset presented in [69] contains inertial data from a smartphone with time-stamped heel strike events and stride lengths, as computed by a foot-mounted inertial system.

The WeAllWalk Dataset

WeAllWalk (https://datadryad.org/stash/dataset/doi:10.7291/D17P46; accessed on 9 June 2021) [16] contains data collected from ten blind and five sighted participants. Nine of the ten blind walkers used a long cane while walking. Two of them repeated the trials using their dog guide. One participant only used her dog guide. The sighted participants and eight of the blind participants walked along six different routes: T1–T4 in one building, and T5-T6 in a second building. The last two blind participants walked on two long routes (T7–T8 for one participant, T9–T10 for the other participant) in the second building. Cumulatively, participants in WeAllWalk walked for 7 miles. Routes were chosen to have a variety of lengths (from 75 to 300 m) and complexities (including multiple 45°, 90° and 180° turns). In some cases, a turn was preceded or followed by a door that had to be opened. The space where data collection took place was partitioned in rectangles ("straight" or "turn") defined on the floor maps; this information is also provided with the dataset. Specifically, for the i-th path, a set $\left\{ P_j^i \right\}$ of consecutive segment endpoint locations (*waypoints*) is provided.

Participants carried two smartphones (iPhone 6s), placed in different locations of their choice on their garments. Each smartphone recorded data (sampled at 25 Hz and timestamped) from its tri-axial accelerometers, gyroscopes, and magnetometers, as well as attitude data produced by the iOS Core Motion framework via Apple's proprietary sensor fusion algorithms. Heel strike times were recorded for each foot using two inertial sensor units clipped to the participants' shoes. Overall, approximately 20,000 heel strikes were recorded.

The time elapsed from the beginning to the end of each route traversal was divided into contiguous intervals, where each interval corresponds to the participant either walking along a "straight" segment in the path, or walking on a "turn" segment. In addition, the dataset provides time-stamped annotations of *features* (136 overall), defined as particular events such as opening a door, bumping into an obstacle, being caught in a door opening, or stopping momentarily. These events are normally associated with anomalous characteristics in otherwise regular inertial data time series.

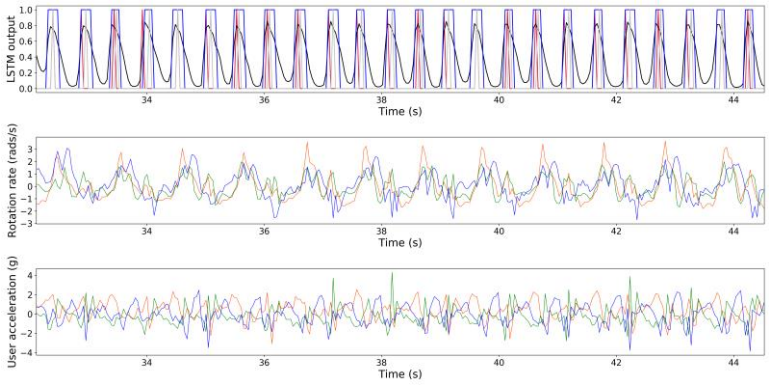
3. Algorithms

In this section, we describe the individual algorithms used by the considered path reconstruction systems. The inertial data of interest from the phones include: *attitude*, defined as the 3-D orientation with respect to a fixed "world" reference frame with Z axis pointing in the direction of gravity; *gyro*, a 3-D vector with angular velocities; *acceleration*, as measured by the 3 axes of the accelerometer; *user acceleration*, which is a 3-D vector with the actual phone acceleration (i.e., with gravity removed). Note that all of these quantities (except for *gyro* and *acceleration*) are obtained using the device's proprietary sensor fusion algorithm from data acquired by the onboard accelerometers and gyroscopes. From this data, we also derive *azimuth* (or heading), which is the angle between the Y axis of the phone frame, projected onto the horizontal plane (defined by X and Y axes of the world frame), and the Y axis of the world frame. It is important to note that *attitude* and, thus, *azimuth* are prone to drift, where, for our purposes, drift can be modeled as a slowly varying bias.

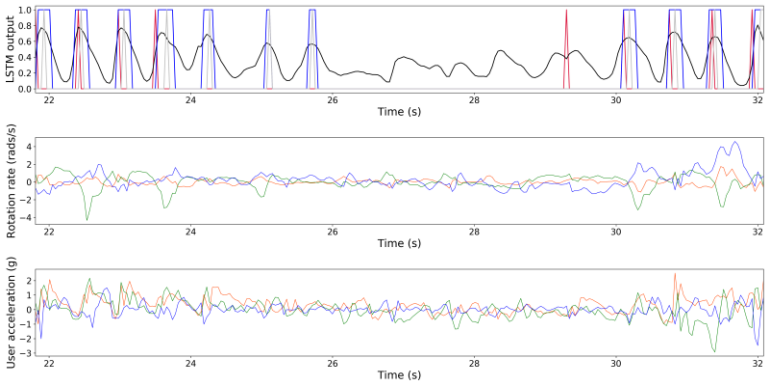
3.1. Step Counting

We implemented a step counting system based on LSTM [70] (a popular type of RNN). Prior work (e.g., [32]) used a bi-directional LSTM, which can increase robustness by considering a whole batch of data at once. This approach would not be appropriate for wayfinding or assisted return application, where timely step detection is necessary (e.g., to constantly track the position of the walker in the route.) We, thus, only considered a regular, uni-directional LSTM for our application. Our LSTM takes in input user acceleration (over 3 axes) and rotation rate (over 3 axes). These vectors are pre-multiplied by the inverse of the attitude matrix, in order to obtain a heading-agnostic reference frame [10]. The data for each axis are normalized to zero mean and unit variance. The LSTM is trained to produce a sequence of values that are close to the desired output, represented by a sequence that is uniformly equal to 0 except for the times of occurrence of a heel strike, when it is set to 1 (more precisely, we transform each impulse into a narrow triangular wave with length of three samples). The LSTM is trained with Keras using 100-samples windows (4 s), and least squares loss. 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. For each sequence of consecutive LSTM output samples that exceed this threshold, we select the midpoint to be the time of estimated heel strike. Our LSTM uses a 2-layer network with hidden unit size of 6. We found, through initial experiments, that the network was deep enough for the task, and adding more layers would increase the risk of overfitting. We used the Adam optimizer and dropout for regularization. Training is performed with batch size of 256 and initial learning rate of 0.005 (decreased by a factor of 2 after 50 epochs) over a total of 64 epochs.

Examples of step detection are shown in Figure 1. Although our step counter works robustly in most conditions, we noticed that the LSTM output exhibits a decaying oscillatory behavior when one stops walking. In addition, sometimes the first one or two steps may be missed when one starts walking.



(**a**)



(**b**)

Figure 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). Examples of overcounts are seen in (**b**) between t = 24 s and t = 26 s. An example of undercount is seen between t = 29 s and t = 30 s.

3.1.1. Stride Length

In order to use step counting for odometry, one needs to define the walker's stride length. A number of techniques have been proposed for stride length estimation from inertial data [46–48], including recent approaches using deep networks [33,34]. We experimented with several of these algorithms, but failed to obtain consistent and accurate results. Thus, we decided instead to use a fixed stride length SL, regressed from the known lengths of the paths traversed in WeAllWalk, and the ground-truth number of steps in each path for each participant in the training set.

3.2. Two-Stage Turn Detection

The trajectory of a walker in an environment characterized by a network of straight corridors crossing at discrete angles (e.g., multiples of 45° or 90°) can in most cases be described as a sequence alternating straight segments with turns, with discrete turning angles. When designing a turn detector to be used by blind walkers, one needs to consider various situations that may trigger a false detection. For example, the walker may stop and rotate their body to re-orient themselves. Walkers may need to swerve when avoiding a perceived obstacle, or when mistakenly veering off a straight path. In order to reduce the likelihood of false detections, we implemented a simple two-stage procedure for robust turn detection. The system (shown in Figure 2) is comprised of an orientation tracker, which returns the discrete angle of the walker's orientation in the horizontal plane; and of a straight-walking (SW) detector, which identifies the time periods in which the user

is walking on an approximately straight path. The idea is that a turn is not expected to occur during a SW segment; whereas any numbers of turns could be detected (correctly or otherwise) outside such intervals. A turn is declared when the user's orientation during a SW segment is different than in the previous SW segment.

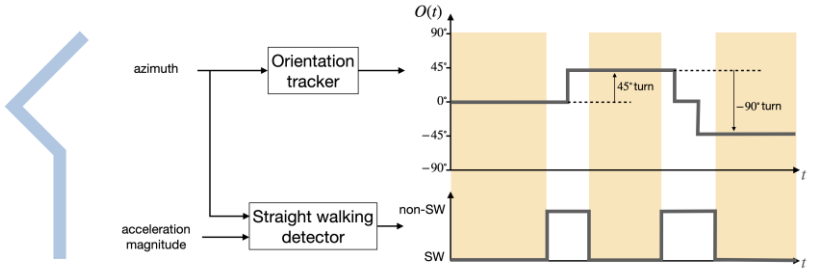


Figure 2. Diagrammatic example of our two-stage turn detector. The blue line (**left**) represents a path taken by a walker, with a 45° (**left**) turn, followed by a $-90°$ (**right**) turn. A turn is detected by comparing the orientation $O(t)$ produced by the orientation tracker between two consecutive straight walking (SW) segments (highlighted in yellow).

3.2.1. Straight Walking (SW) Detector

Our straight walking (SW) detection system is designed to identify time intervals during which the user walks "regularly" on a straight path. SW detection is performed using a GRU, which is a simple type of recurrent neural network [71]. Our GRU processes a pair of signals: the azimuth angle and the user accelerometer magnitude (previously smoothed by a Gaussian filter with $\sigma = 15$). The GRU is trained on data samples recorded within straight segments (after removing *feature* sub-segments), as these are considered to be representative of SW segments. We manually annotated sub-segments at the beginning and at the end of each trial, when the user was known to be standing still, and removed them from the data labeled as SW. Data from these sub-segments, as well as data from *feature* sub-segments and data from turn segments, are labeled as non-SW. The GRU is tasked with predicting the label value with a delay of approximately 1 s (30 samples). We found this to be an acceptable trade-off between timeliness (which calls for small delay) and the need to see enough forward data to correctly predict the label after a transition. The system was trained using Keras, using 150-samples windows. Other training parameters include: GRU hidden unit size of 32; drop-out rate of 0.4; 3 training epochs, batch size of 2048. An example of SW detection using our GRU system is shown in Figure 3.

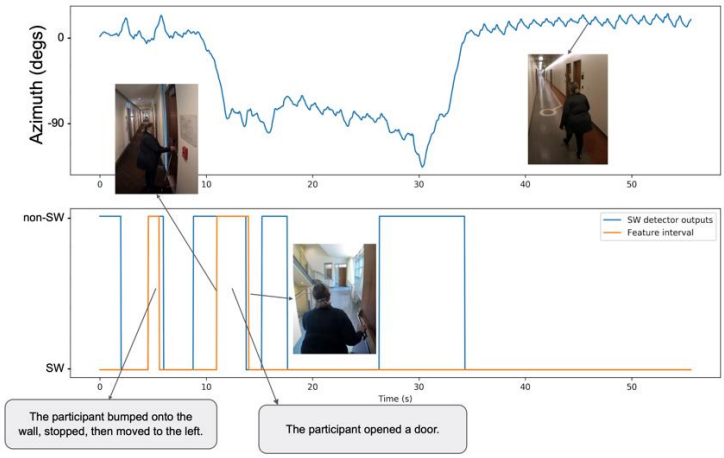


Figure 3. Example of SW segment detection using our GRU system. Top: Azimuth signal; Bottom: output of SW detector (blue line), shown together with the segments marked as "features" in WeAllWalk (orange line).

3.2.2. Orientation Tracker

The goal of this module is to track the walker's orientation from azimuth data in the face of slowly changing drift. We assume that the walker is at each time t in one of N fixed orientations $\left\{O^k(t) = k{\cdot}360^\circ/N\right\}$, where N = 8 or 4 (orientations at multiples of 45° or 90°, respectively). The sequence of orientations is modeled as a Markov chain. The azimuth measurement, $\theta(t)$, is assumed to be affected by drift, $d(t)$, which is modeled as a random walk [72], with additional white Gaussian noise. We use the standard Gauss–Markov hypothesis to write:

$$\begin{aligned} p(\theta(t)|O(t),d(t),\theta(1:t-1)) &= p(\theta(t)|O(t),d(t)) = \mathcal{N}\left(O(t)+d(t),\sigma_\theta^2\right) \\ p(d(t)|d(t-1),\theta(1:t-1)) &= p(d(t)|d(t-1)) = \mathcal{N}\left(d(t-1),\sigma_d^2\right) \\ P)O^k(t)|O^m(t-1),\theta(1:t-1)) = P(O^k(t)|O^m(t-1)) &= \begin{cases} q, & m=k \\ (1-q)/2, & |m-k| = 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

In the equations above, $\theta(1:t-1)$ represents the sequence of azimuth measurements before time t. $\mathcal{N}(\mu,\sigma^2)$ is a normal distribution, while q represents the prior probability that the user's orientation does not change in two consecutive time instants. Note that turns $O^m(t-1) \to O^k(t)$ by more than the prescribed interval $360^\circ/N$ (i.e., $|m-k| > 1$) are not allowed in this model. For example, if $N = 8$, a turn by 90° would be tracked as two consecutive turns by 45°.

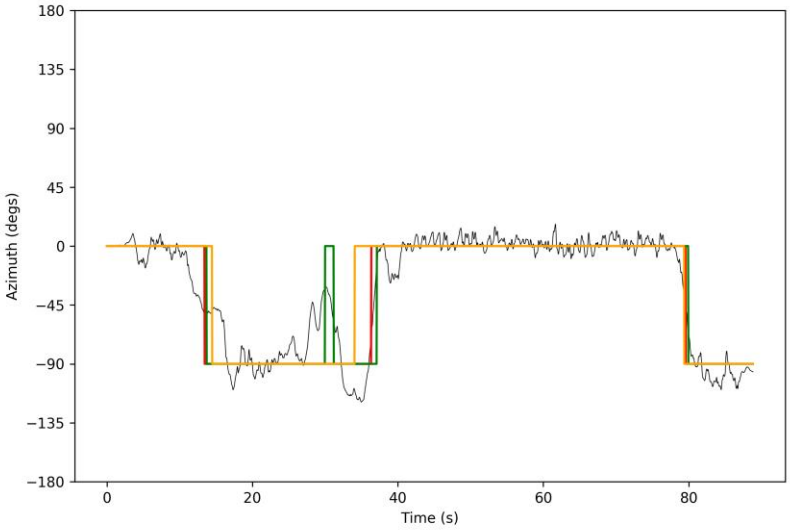
We use a Mixture Kalman Filter (MKF [73]) to compute the posterior distribution of discrete orientations $P(O^k(t)\big|\theta(1:t))$ and of drift $p(d(t)|\theta(1:t))$ at each time. The MKF algorithm maintains a list of orientation sequences, with a Kalman filter associated with each sequence. At each time instant, a new orientation is added to each sequence by sampling from the posterior distribution $P(O(t)|\theta(1:t))$. Our MKF system maintains a set of 50 Kalman filters, which go through a standard process of resampling and rejuvenation [73]. The walker's orientation at time t is taken to be the maximizer of $P(O(t)|\theta(1:t))$. The parameters σ_θ, σ_d, and q are learned from the training data by finding (via grid search) the minimum of the weighted sum of overcount and undercount rates of the two-stage turn detector, as defined later in Section 4.3. Note that we give a larger weight (equal to 2.5) to the undercount rate, because we noted that turn undercounts often affect the reconstructed path more than turn overcounts, possibly because two consecutive incorrectly detected turns with opposite angles compensate for each other.

3.2.3. Turn Detection

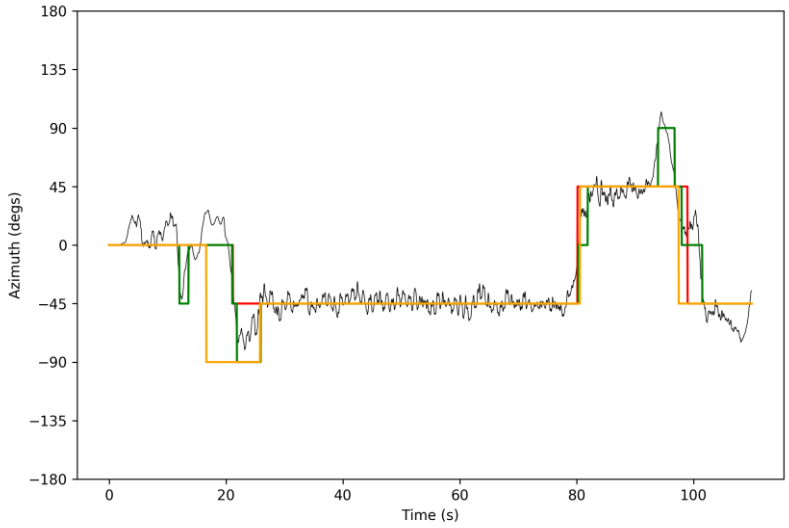
In our two-stage system, turn events are detected when the discrete walker's orientation $O(t)$ in a SW segment is different from that in the previous SW segment. Note that the orientation of a walker is not expected to change in a SW segment. However, occasionally a change in orientation is observed, typically at the beginning of the segment. This may happen if the MKF system reacts with some delay to a change in orientation associated with a turn. To remove the risk of these situations, we compute the mode of the orientations in each SW segments, and compare it with the mode of orientations in the previous SW segment for turn detection.

The orientation resolution (45° or 90°) to be chosen for the MKF orientation tracker depends on the specific environment. Although most corridor networks intersect at ±90°, in some cases intersections at multiples of ±45° should be accounted for. For example, in the WeAllWalk dataset, 13% of all turns are ±45° turns. Note that a change of orientation by 180° or (when using orientation resolution of 45°) by ±90°, is often detected as a close sequence of orientation changes by ±90° (or ±45°). This is usually not a problem in our two-stage strategy, where turns are declared only by comparing the orientation in two consecutive SW segments. For example, if a sequence of two 45° turns is detected during a non-SW segment, our system will declare that one 90° turn occurred (see Figure 2).

Sample results from our two-stage turn detection system are shown in Figure 4 (SW detection is not shown in these figures). Note that in Figure 4a, at around t = 30 s, a close sequence of orientation changes (left then right) as measured by the MKF was triggered as a consequence of the walker veering off the straight path (note the large variation in azimuth angle). As this happened outside of a SW period, these incorrectly detected turns were rejected by the two-stage system. A similar situation can be observed in Figure 4b, where the MKF was set to measure orientations at multiples of 45°.



(**a**)



(**b**)

Figure 4. Example of two-stage turn detection. Black line: azimuth signal. Green line: walker's discrete orientation as tracked by MKF. Orange line: walker's discrete orientation as obtained by integrating the turns detected by the two-stage system. Red line: ground-truth walker's discrete orientation. The orientation resolution of the MKF was 90° (**a**) or 45° (**b**).

3.3. RoNIN

RoNIN [10] is a deep network end-to-end odometry algorithm that was shown to outperform comparable systems in challenging datasets. It uses acceleration and gyro data, along with attitude data, to compute velocity vectors which, integrated over time, produce the user's position. (We do not utilize the RoNIN body heading network since we are only interested in reconstructing trajectories). RoNIN uses a heading-agnostic reference frame. In practice, the 3-D vectors of acceleration and gyro are pre-multiplied by the inverse of the attitude matrix, so that they effectively are defined on a fixed "world" reference frame with z axis pointing in the direction of gravity. Although this reference frame normalization was shown to enable robust odometry, any drift accumulated in the computed attitude also reflects in the velocity vectors produced in output.

In order to apply RoNIN to the WeAllWalk inertial data, it is first necessary to upsample (via linear interpolation) the data from the original acquisition rate of 25 Hz to 200 Hz, which is the rate at which sensor data were collected for RoNIN training. We used the open source implementation provided by the authors (https://github.com/Sachini/ronin; accessed on 9 June 2021) and chose the RoNIN resnet18 architecture. Since the data used for training RoNIN came from sensors that are different from those in the iPhones used in WeAllWalk, results may be different than expected. As a simple customization, we regressed a constant scaling factor to minimize the discrepancy between the magnitude of the velocities computed by RoNIN and of the ground-truth velocities in WeAllWalk. Least squares regression was used to compute an appropriate scaling factor (found to be equal to 1.27) to be applied to the velocity vectors produced by RoNIN.

We also considered a more extensive customization of RoNIN, by fine-tuning the network on WeAllWalk data. Note that RoNIN requires ground-truth values for the walker's location and orientation at all times, defined with respect to the world reference frame used to define the phone attitude; whereas WeAllWalk only contains the times at which each walker crossed specific waypoints. To generate the missing data, we interpolated the walker's location between straight segments' waypoints assuming constant velocity, and assumed that, in a given segment, walkers maintained a constant orientation parallel to the corridor. In order to recover the walker's orientation with respect to the world reference frame, we rely on the orientation estimated at each time by the original RoNIN network. We then fine-tuned the RoNIN network (resnet18 architecture) for two epochs, with minibatch size set to 128, using the Adam optimizer with learning rate of 0.0001. We only considered this customization for the blind walkers, given that the original RoNIN was already trained on sighted walkers. Indeed, we verified that fine-tuning the algorithm on the sighted walkers in WeAllWalk did not lead to noticeable improvements.

3.4. Particle Filtering

When a map of the environment is available, the constraint of wall impenetrability can benefit path reconstruction. Bayes filtering (in particular, particle filtering [9]) is typically used in these situations, as it can easily incorporate the wall impenetrability constraint by postulating an appropriate form for the joint distribution of the state s being tracked. For example, if the state contains the location of the walker, the joint probability $P(s(t), s(t+\Delta t))$, where Δt is the sampling period, is set to be 0 if $s(t)$ and $s(t+\Delta t)$ identify two locations across a wall from each other. Particle filtering is a particular form of Bayes filtering wherein the posterior distribution of the state is represented by a set of samples (particles). Statistics such as the posterior mean can be obtained by simply computing the mean location of these particles.

In our implementation, the particle filter receives in input a sequence of velocity vectors $\boldsymbol{v}(t) = ||\boldsymbol{v}(t)|| \cdot [\cos\theta(t), \sin\theta(t)]^T$, which are generated either by the RoNIN algorithm (in its original or fine-tuned version, and subsampled to a rate of 25 Hz), or by forming a vector in the direction of the azimuth, as measured by the phone, with length equal to the stride length considered, divided by the duration of the current step, as measured by

our step counter. The state being tracked has two components: the current location of the walker, P_t, and the current drift angle, d_t. State evolution is modeled as:

$$\begin{aligned} P(t+\Delta t) &= P(t) + (||\boldsymbol{v}(t)|| + n_v) \cdot [\cos(\theta(t) + d(t) + n_\theta), \sin(\theta(t) + d(t) + n_\theta)]^T \cdot \Delta t \\ d(t+\Delta t) &= d(t) + n_d \end{aligned} \quad (2)$$

Here, n_v, n_θ, and n_d represent mutually uncorrelated zero-mean Gaussian white noise. We set the standard deviations of n_θ and n_d to 0.01 rads and 0.005 rads, respectively, where these values were found empirically through preliminary tests. We noted that the system performance is rather sensitive to the choice of the standard deviation σ_v of n_v, and, thus, decided to determine this value systematically through grid search on training data. We assume that the location of the starting point in the map is known, and perform an initial alignment procedure (similar to [10]) by matching the path reconstructed in the initial few seconds with the orientation of the corridor where the starting point is located. The particle filter uses 1000 particles.

A problem with Bayes filtering as applied for path reconstruction is that it often produces bi-modal posterior state distributions (see Figure 5). In these situations, the posterior mean often falls in areas of low probability (in-between the modes). In practice, this means that the mean location of the particles may fall in a location that appears to break the wall non-penetrability constraint. Rather than the posterior mean, one could consider the modes (maximizers) of the posterior distribution, which are relatively immune to these artifacts. For example, Fusco and Coughlan [63] used a kernel density estimator to find the mode of the state posterior distribution. We took a different route and used the mean shift algorithm [74], a technique based on gradient ascent to find the modes of a distribution expressed as a set of samples. Our Particle Filtering–Mean Shift (*PF-MS*) algorithm thus estimates the walker's location as the location of the highest mode identified by Mean Shift. If one is interested only in the reconstructions of the whole trajectory (and not in instantaneous location estimation), a variation of PF-MS can be used. Specifically, one can track the modes of the posterior state distribution, by associating a mode at time $t + \Delta t$ with the mode at time t that shares the largest number of supporting particles. At the end of the trajectory, the highest mode of the final posterior distribution is found, then the chain of pairwise associated modes in the previous time instants is reconstructed. We named this "global" strategy *PF-MS-G*. We used the python mean shift implementation from sklearn, setting the bandwidth to 5 m, a value found empirically in preliminary tests.

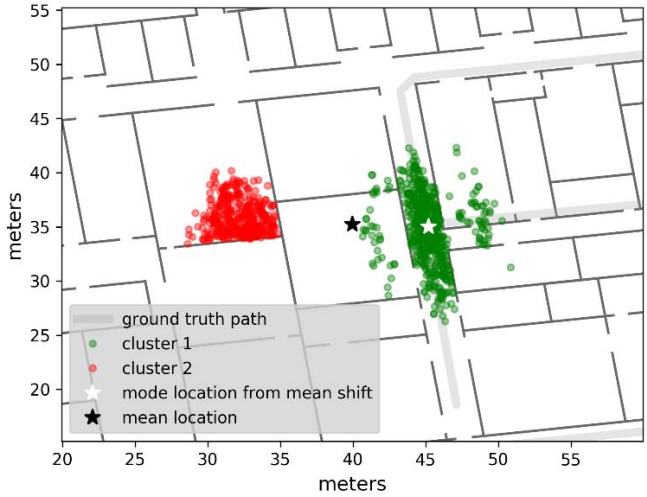


Figure 5. The set of particles at a certain time during tracking. Note that the particles are distributed in two main clusters (shown in different colors). The mean particle location (posterior mean), shown by a black star, is in an incorrect position. The highest mode of the distribution, as found by mean shift, is shown by a white star.

The digital maps of the building walls were obtained by tracing the original maps provided as part of WeAllWalk using the SIM web application (https://sim.soe.ucsc.edu; accessed on 9 June 2021) [75]. We assumed that all doors in the buildings were open, meaning that the estimated paths could potentially go through such doors into rooms (even though the participants only walked along the buildings' corridors.)

4. Experiments

4.1. Training and Test Modalities

The algorithms discussed in Section 3 contain a number of trainable parameters, and specifically: the parameters of the LSTM used for step counting, along with the threshold on the output of LSTM and the stride length (Section 3.1); the values for σ_θ, σ_d, and q in the MKF used for orientation tracking (Section 3.2.2); the parameters of the GRU used for SW detection (Section 3.2.1); and the value of σ_v for the particle filter. In addition, as discussed in Section 3.3, the RoNIN algorithm was fine-tuned for the blind participants.

Since the walking characteristics of blind persons using a long cane or a dog guide are expected to be different [76], we stratified the results by community of long cane users (denoted by the modifier *:LC*) and dog guide users (*:DG*). We considered the following training and test schemes. Note that the system used to test any given walker was never trained on data from that walker.

- **Train on Sighted (TS).** All parameters are computed using data from the five sighted walkers in WeAllWalk. The system is tested on the two community of blind users (*TS:LC*, *TS:DG*). This may be representative of a situation in which a system designed for sighted walkers is used by blind walkers, without any customization.
- **Train in same Community (TC).** In this case, the system tested with long cane users, dog guide users, and sighted users, was trained with data from walkers in the same community. We used the *Leave-One-Person-Out* cross–validation policy [77]: each participant was tested with the system trained on data from all other participants in the same community. This training modality allows us to test the hypothesis that walking characteristics may be different between communities of users. If this hypothesis is true, one may expect that training the system on the same community of users who are meant to use it should give improved results. In addition to the two communities of blind participants (*TC:LC*, *TC:DG*), for this modality only we also present results for the sighted participants (*TC:S*). The latter quantity may be representative of a conventional system, trained and tested on sighted users, and can be used as a benchmark. Of note, only three walkers in WeAllWAlk used a dog guide, and, thus, each training set in the *TC:DG* modality contain data from two walkers only.
- **Train on All (TA).** All available data in WeAllWalk were used for training, using the Leave-One-Person-Out cross–validation policy (*TA:LC*, *TA:DG*). For example, a long cane user is tested with a system trained with data from all sighted participants, all dog cane users, and all 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.

4.2. Step Counting

We considered two different error metrics for the step counter. The first metric (*SC-Error 1*) is a variation of the metric with the same name in [16]. 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. For the second metric, SC-Error 2 [16], 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 (Section 3.1) normally results in an increase in the UC rate and a decrease in the OC rate. This is shown in Figure 6, where we plotted the UC rate vs. OC rate as a function of S. For the test results shown in Table 1, the threshold S is set to a value that equates the OC and UC rates of SC-Error 2 measured in the training data. The values of S thus computed, averaged over all cross-validation rounds, are also shown in Table 1, along with the average value of stride length SL regressed from the training data as discussed in Section 3.1.1.

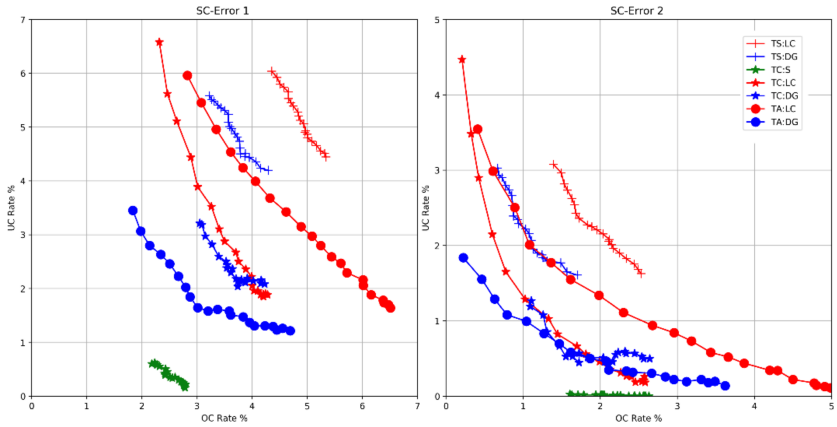


Figure 6. UC rate vs. OC rate curves as a function of the threshold S on the LSTM output.

Table 1. Undercount (UC) and overcount (OC) rates for our step counter, along with the mean threshold S and stride length SL. For each community of blind walkers (LC, DG), the pair (UC rate, OC rate) with the smallest value of their sum is shown in boldface.

	SE-Error 1		SE-Error 2		Mean S	MeanSL (m)
	UC Rate %	OC Rate %	UC Rate %	OC Rate %		
TS:LC	10.58	2.51	8.20	0.13	0.78	0.74
TS:DG	8.74	2.08	6.75	0.10	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:DG	4.99	2.15	3.23	0.39	0.68	0.62
TA:LC	4.06	3.80	1.42	1.17	0.56	0.62
TA:DG	**2.29**	**2.55**	**0.83**	**1.08**	0.55	0.62

4.3. Turn Detection

To quantitatively assess the performance of our two-stage turn detector (Section 3.2), we compare the sequence of detected turns against the ground-truth sequence of turns in each path using Levenshtein distance. Specifically, we find the longest ordered matching subsequences within the two turns sequences, then report errors in terms of overcounts (detected turns that were not matched in the ground-truth sequence) and undercounts (ground-truth turns that were not matched in the detected sequence). Note that, in the case of a 45° ground-truth turn, a 90° turn detector would either not detect the turn, or detect it as a 90° turn, where both cases represent an incorrect result. To simplify analysis of the results, we simply removed the 45° turns and any associated 90° turn detection from the analysis of the 90° turn detector. When assessing the 45° turn detector, we split all ground-truth 90° turns as well as all measured 90° turns into two consecutive 45° turns. The number of turn undercounts and overcounts was divided by the total number of ground-truth turns to obtain the undercount (UC) and overcount (OC) rates, which together define the turn detection (TD) error. Results are shown in Table 2.

Table 2. Undercount (UC) and overcount (OC) rates for our turn detector (45° and 90° turns). For each community of blind walkers (LC, DG), the pair (UC rate, OC rate) with the smallest value of their sum is shown in boldface.

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

4.4. Path Reconstruction

4.4.1. Evaluation Metrics

Each reconstructed trajectory is compared with the ground-truth path. WeAllWalk does not provide detailed information of the walkers' location at each time, but only the timestamps $\left\{t_j^i\right\}$ indicating when a walker reached individual waypoints. We will make the simplifying assumption that walkers were located in the middle of the corridor width when transitioning between segments through each such waypoint.

Given a trajectory estimated for a walker in a path, $P^i(t)$, we use the locations at the waypoint timestamps, $\left\{P^i\left(t_j^i\right)\right\}$ to first *align* this trajectory with the ground-truth path. This standard procedure [78] is necessary because the reference frame used to represent the trajectory is undefined. Specifically, we find (using Procrustes analysis) the rotation and translation that minimizes the mean squared distance between the locations $\left\{\overline{P}_j^i\right\}$ (the ground-truth location of the waypoints) and $\left\{P^i\left(t_j^i\right)\right\}$. After alignment, we evaluate the goodness of the estimated path using three different metrics. The first metric considered is the RMSE of estimated waypoint locations:

$$\mathrm{RMSE_{wp}} = \sqrt{\frac{1}{N}\sum_j ||\overline{P}_j^i - P^i\left(t_j^i\right)||^2} \tag{3}$$

where N is the number of waypoints in the path. For the remaining metrics, we consider a sampling of the estimated trajectory into N_e^i points $\left\{Q_m^i\right\}$ with a uniform inter-sample distance of 1 m. Likewise, we sample the segments joining consecutive waypoints at intervals of 1 m, resulting in N_{gt}^i points $\left\{\overline{Q}_n^i\right\}$ representing the path. We then compute the Haussdorff distance between these two sets of points, as well as their (weighted) average Haussdorff distance [79]:

$$\mathrm{Hauss} = \max\left(\max_m\left(\min_n\left(||Q_m^i - \overline{Q}_n^i||\right)\right), \max_n\left(\min_m\left(||Q_m^i - \overline{Q}_n^i||\right)\right)\right) \tag{4}$$

$$\mathrm{avHauss} = \frac{1}{2}\left(\sqrt{\frac{1}{\mathrm{N_e}}\sum_m \min_n\left(||Q_m^i - \overline{Q}_n^i||^2\right)} + \sqrt{\frac{1}{\mathrm{N_{gt}}}\sum_n \min_m\left(||Q_m^i - \overline{Q}_n^i||^2\right)}\right) \tag{5}$$

The Haussdorff distance penalizes any large (possibly episodic) discrepancy between the estimated trajectory and the ground truth. The average Haussdorff is a more lenient measure, that penalizes consistent biases. These two metrics allow us to evaluate the goodness of the estimated trajectory in its entirety (and not just at waypoints). Figure 7 illustrates the three considered metrics.

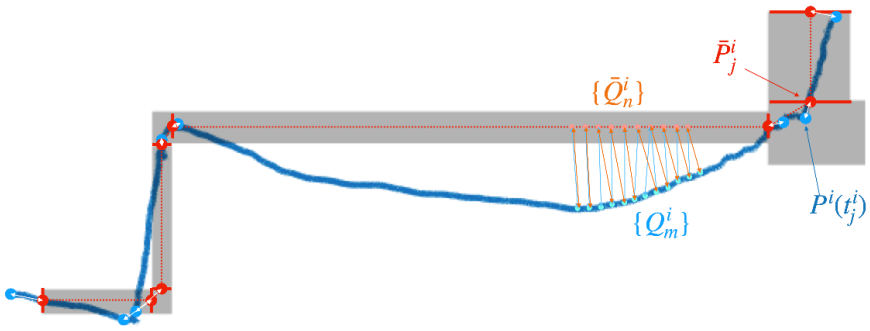


Figure 7. An illustration explaining the metrics used to evaluate the quality of an estimated trajectory $P^i(t)$ (shown as blue line after alignment). The gray shape shows the location of the corridors along the path, with individual segment separation shown in red. Waypoints ($\overline{P}_j^i$) are shown as red dots, while the estimated walker's locations at waypoint timestamps ($P^i\left(t_j^i\right)$) are shown as blue dots. The distances $||\overline{P}_j^i - P^i\left(t_j^i\right)||$ are shown by white arrows. The uniform distance sampling of the segments joining consecutive waypoints ($\left\{\overline{Q}_n^i\right\}$) is shown by orange dots, while that of the estimated trajectory ($\left\{Q_m^i\right\}$) is shown by light blue dots. The associated distances $\min_m\left(||Q_m^i - \overline{Q}_n^i||\right)$ and $\min_n\left(||Q_m^i - \overline{Q}_n^i||\right)$ are shown with orange and light blue arrows, respectively.

4.4.2. Map-Less Path Reconstruction

We considered the following algorithms for map-less path reconstruction (see Figure 8):

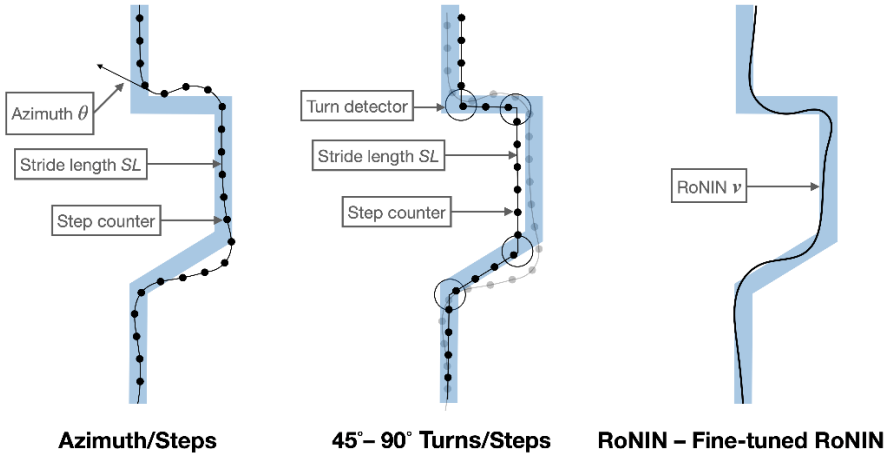


Figure 8. Diagrammatic examples of the algorithms used for map-less path reconstruction. The blue line represents the path taken by the walker. The black line represents the estimated path. Dots represent heel strikes; circles represent turns.

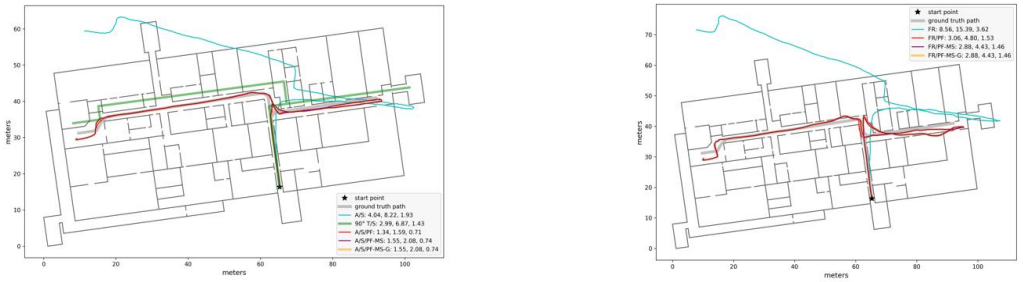
- **Azimuth/Steps (A/S)**: At each detected step, a displacement vector is defined with length equal to the step stride, and with orientation equal to the azimuth angle θ as provided by the phone;
- 45°–90° **Turns/Steps (T/S)**: At each detected step, a displacement vector is defined with length equal to the step stride, and with orientation equal to the output of our two-stage 45° or 90° turn detection method;
- **RoNIN (R)–Fine-tuned RoNIN (FR)** (see Section 3.3).

Reconstruction errors are shown for the three considered metrics in Table 3. Note that fine-tuned RoNIN was only considered for blind walkers (LC and DG). Examples of map-less path reconstruction using fine-tuned RoNIN, Azimuth/Steps, and 90º Turns/Steps, are shown in Figure 9. Although the reconstruction errors are computed after alignment as

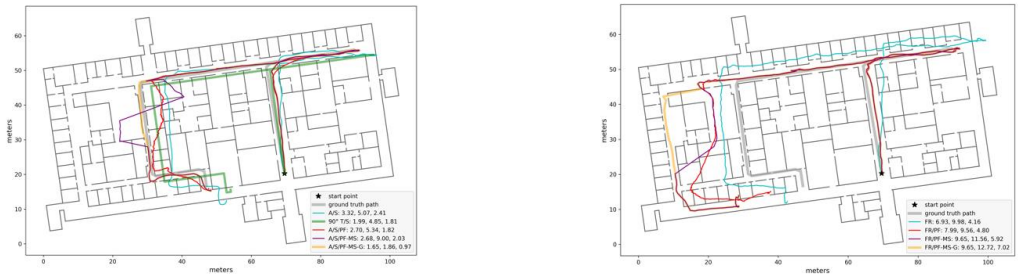
described in Section 4.4.1, the paths have been re-aligned on the underlying map in the figure for easy comparison with the map-assisted reconstructed paths.

Table 3. Reconstruction errors ($RMSE_{wp}$, *Hauss*, *avHauss*) using the map-less path reconstruction algorithms described in Section 4.4.2. Units of meters. For each community of walkers (S, LC, DG), the smallest error values for each metric are shown in boldface.

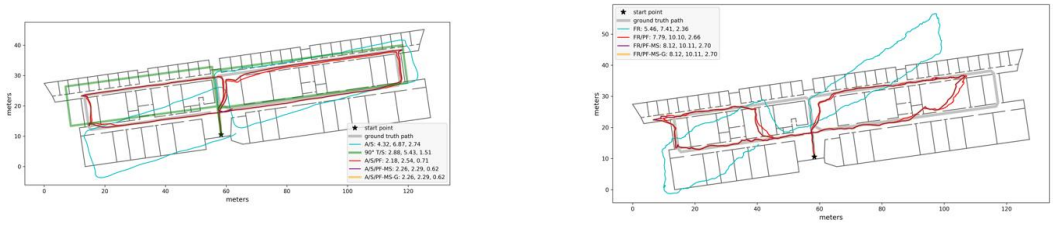
	A/S			45° **T/S**			90° **T/S**			**R**			**FR**		
TS:LC	9.43	14.68	4.90	9.17	14.48	4.70	9.03	13.90	4.63	5.43	8.56	2.93	–	–	–
TS:DG	5.81	8.97	3.30	5.64	8.67	2.82	4.94	**7.50**	2.75	5.66	8.55	2.84	–	–	–
TC:S	4.45	7.06	2.39	3.96	6.12	1.89	**3.93**	**5.97**	**1.88**	4.26	6.75	2.39	–	–	–
TC:LC	3.85	6.21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



(**a**)



(**b**)



(**c**)

Figure 9. Examples of path reconstruction from the TA:LC training/test modality. Left: Map-less: A/S, 90° T/S; Map-assisted: A/S-PF, A/S-PF-MS, A/S-PF-MS-G. Right: Map-less: FR; Map-assisted: FR-PF, FR-PF-MS, FR-PF-MS-G. Reconstruction errors for the three metrics (RMSEwp, Hauss, avHauss) are shown (in units of meters) in the legend of each figure. (**a**–**c**) refer to different buildings.

4.4.3. Map-Assisted Path Reconstruction

We experimented with feeding the particle filter with data generated by three different algorithms: Azimuth/Steps (A/S), RoNIN, and, for blind walkers, fine-tuned RoNIN. The Turns/Steps algorithm was shown to give comparatively poor results in this case. Table 4 shows the results, for the three metrics considered, using particle filtering (PF), as well as particle filtering with mean shift mode selection in the "instantaneous" mode (PF-MS) and in the "global" mode (PF-MS-G; see Section 3.4). Sample reconstructed paths are shown in Figure 9.

Table 4. Reconstruction errors ($RMSE_{wp}$, *Hauss*, *avHauss*) using the map-assisted path reconstruction algorithms described in Section 4.4.3. Units of meters. For each community of walkers (S, LC, DG), the smallest error values for each metric are shown in boldface.

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

5. Discussion

5.1. Step Counting

The data in Table 1 and the curves in Figure 6 clearly show how step counting accuracy is affected by the community of walkers used to train the system. For example, when testing with long cane users a step counter trained on sighted walkers (TS:LC), the sum of undercount and overcount rates was found to be 13.09% (SC-Error 1) or 8.33% (SC-Error 2). However, when the system was trained only with other long cane users (TC:LC), these numbers reduce to 6.49% and 1.87%, respectively. Similar observations can be drawn from the tests with dog guide users, for whom the best results were obtained when training on all available data (TA:DG). A possible cause for the worse performance of TC:DG with respect to TA:DG is the small number of dog guide users in WeAllWalk (only two users in the training set of each cross-validation round).

The average threshold S on the output of the LSTM as learned from within-community training data is substantially larger for sighted walkers (0.78) than for dog guide (0.68) or 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 stride lengths learnt within-community are also larger for sighted walkers (0.74 m) than for dog guide users (0.62 m) or long cane users (0.55 m). This is not surprising considering that, in general, dog guide users walk faster and more confidently than long cane users, as they do not need to probe the space ahead with the cane and can rely on their dog guide to lead them along a safe trajectory. Comparison with [16], which also presents results for various step counting algorithms as applied to WeAllWalk, shows that use of a LSTM leads to improved results. For example, the lowest value for SC-Error 1 (measured as the sum of UC and OC rates) for long cane users was found to be 7.8% in [16] (vs. 6.5% with our system, see Table 1). For the same community of users, the minimum SC-Error 2 found in [16] was 4.8%, vs. 1.9% with our system.

5.2. Turn Detection

Remarkably, Table 2 shows no undercounts or overcounts for sighted walkers (TC:S). This suggests that these participants tended to walk on straight lines, without large sway patterns that could generate false positives. Errors were generally higher for the 45° than for

the 90° turn detectors. These results should be evaluated keeping in mind that even a single missed turn, or a single false positive, could potentially lead to large path reconstruction errors. For long cane users, training with all available data (TA:LC) gave substantially better results than training only with data from sighted walkers (TS:LC). No such large discrepancies across training modalities were observed when testing with dog guide users, for whom the best results were obtained for within-community training (TC:DG). These results are vastly superior than those observed in [16], where turns were computed on WeAllWalk data using the algorithm described in [38]. In that case, the accumulated error (UC rate + OC rate) 90° TD-Error was found to exceed 50%.

5.3. Map-Less Path Reconstruction

The data in Table 3 shows that the smallest reconstruction errors were measured for the 90° Turns/Steps algorithm (although for dog guide users, a similar small error for the avHuass metric was obtained with fine-tuned RoNIN). The 45° Turns/Steps algorithm performed only marginally worse than the 90° case. Remarkably, the best training modality for long cane users (within-community training, TC:LC), gives similar error as the best case for sighted walkers. However, when testing with long cane user using a system trained with sighted walkers, very poor results were obtained. For example, the $RMSE_{wp}$ error for the 90° Turns/Steps system, which is 3.46 m for within-community training (TC:LC), jumps to 9.03 m when training on sighted walkers (TS:LC). For dog guide users, the best results were obtained by training over all available data. Both RoNIN and the simpler Azimuth/Steps algorithms are affected by drift, and produced comparable results.

One notable exception is for long cane users when the system was trained with sighted walkers (TS:LC). In this case, RoNIN gave substantially better results than all other methods (though still worse than TC:LC). A likely reason for this can be found in the different average stride length between sighted and long cane users (see Section 5.1), which may cause incorrect reconstructed path lengths for TS:LC. RoNIN, which does not rely on stride lengths, may provide more accurate velocity measurements in this case. The data also shows that fine-tuning the RoNIN network did not result in improved path reconstruction performance.

5.4. Map-Assisted Path Reconstruction

The best results for the community of blind walkers were obtained with the azimuth/steps (A/S) algorithm, processed by the PF-MS-G filter. For the sighted walkers, the best results were obtained with RoNIN, still processed by PF-MS-G, although these last results were only marginally better than using the A/S algorithm. It appears that the strong wall impenetrability constraint was able to limit the effect of azimuth drift. In general, errors for map-assisted reconstruction were substantially lower than for the map-less case. As in the prior cases, training the system over sighted walkers was shown to give poor results when tested with long cane users and, to a lesser extent, with dog guide users. Although the best results were obtained with the PF-MS-G algorithm, use of the mean shift clustering was not shown to give consistently better results overall.

Figure 9 provides some insight into the behavior of the reconstruction algorithms. In Figure 9a, both original paths (A/S, fine-tuned RoNIN) were grossly incorrect due to orientation drift. Particle filtering tracked and removed drift, and correctly reconstructed the paths. In the case of Figure 9b, poor velocity estimation led to reconstructed segments that were too short (Azimuth/Steps) or too long (fine-tuned RoNIN). In both cases, particle filtering found incorrect paths through open doors. PF-MF-G was able to correctly reconstruct most of the path for the Azimuth/Steps case, but not for the fine-tuned RoNIN case. The trajectory in output of either system in the case of Figure 9c was affected by both drift and incorrect velocity estimation. Although particle filtering was for the most part able to successfully correct the trajectory in the Azimuth/Steps case, it produced a poor reconstruction for fine-tuned RoNIN.

We note that the set of path reconstruction algorithms considered in this work includes inertial-based odometry algorithms used in prior research for blind walkers. For example, [68] used a PDR-based on step counting coupled with orientation (equivalent to our A/S system), while [67] additionally implemented a particle filter (equivalent to our A/S-PF system).

6. Conclusions

We have presented a detailed study on the use of two odometry systems: a simple PDR based on step counts and orientation estimation, and a deep learning algorithm (RoNIN), to reconstruct paths taken by blind walkers using a long cane or a dog guide, as represented in the WeAllWalk dataset. We have considered both the map-less case and the map-assisted case. For the map-less case, we have introduced a two-stage system capable of robustly detecting turns at multiples of 45° or 90° degrees, combined with an RNN-based step counter with learned fixed stride length. For the map-assisted case, we employed a standard particle filtering algorithm, with the addition of a posterior distribution mode identification module. Compared with prior work that explored inertial odometry for use by blind pedestrians (e.g., [18,67,68]), our study includes a variety of algorithms for both the map-assisted and map-less case, and reports the results of extensive quantitative evaluations using appropriate metrics on a representative data set with blind walkers (WeAllWalk). Although it is possible that other sensing modalities (e.g., visual odometry [63] or BLE-based positioning [65]) may achieve higher localization accuracy, inertial odometry has clear practical advantages, as it requires no additional infrastructure, and may function even if the user keeps the smartphone in their pocket while walking.

Our study has shown that, for the same algorithm, the choice of the community of walkers used for training the algorithm's parameters is critical. Systems trained with sighted walkers consistently gave poor results when tested with long cane users and, to a lesser extent, with dog guide users. However, when the training set contained data collected from these communities, results improved substantially, and were in fact comparable to the best results obtained when testing with sighted walkers. Our results also showed that our simple Turns/Steps PDR produced better results than the more sophisticated RoNIN in the map-less case, even when the latter was fine-tuned to better model walking patterns of blind individuals. For the map-assisted case, the best results were found when the particle filter was fed with data from the simple Azimuth/Steps algorithm. It should be noted, however, that participants in WeAllWalk kept their smartphones in a fixed location on their garments. Had they changed the orientation of their phone (e.g., to pick up a call, or by repositioning the phone in a different pocket), it is likely that this change in phone orientation would have negatively affected the results. In these situations, an algorithm such as RoNIN, which was trained to correctly identify the user velocity independently of the phone orientation, possibly combined with a mechanism to reduce the effect of drift, could provide more robust position tracking.

The choice of the minimum turn angle for a Turn/Steps PDR depends on the specific environment considered. Although the layout of most buildings is made by corridor networks intersecting at 45° or 90° degrees, there may be situations that call for a finer angular interval. Although, in principle, this could be achieved simply by increasing the cardinality of the state tracked by the MKF (Section 3.2.2), in practice this could result in increased false positives. In addition, we have found (as mentioned in Section 3.2.3) that turns of 90° degrees are often detected as a sequence of two 45° turns. We expect that a similar behavior would be amplified in the case of a finer angular resolution.

Author Contributions: Conceptualization, R.M., P.R. and F.E.; methodology, R.M., P.R. and F.E.; software, P.R. and F.E.; validation, F.E. and P.R.; formal analysis, P.R. and F.E.; investigation, R.M., P.R. and F.E.; resources, R.M.; data curation, P.R. and F.E.; writing—original draft preparation, R.M.; writing—review and editing, R.M., F.E. and P.R.; visualization, P.R. and F.E.; supervision, R.M.; project administration, R.M.; funding acquisition, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: Research reported in this report was supported by the National Eye Institute of the National Institutes of Health under award number R01EY029260-01. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [https://datadryad.org/stash/dataset/doi:10.7291/D17P46, accessed on 9 June 2021].

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Guerreiro, J.; Ahmetovic, D.; Sato, D.; Kitani, K.; Asakawa, C. Airport Accessibility and Navigation Assistance for People with Visual Impairments. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland, UK, 4–9 May 2019; pp. 1–14.
2. Murata, M.; Ahmetovic, D.; Sato, D.; Takagi, H.; Kitani, K.M.; Asakawa, C. Smartphone-Based Indoor Localization for Blind Navigation across Building Complexes. In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), Athens, Greece, 19–23 March 2018; pp. 1–10.
3. Spachos, P.; Plataniotis, K.N. BLE Beacons for Indoor Positioning at an Interactive IoT-Based Smart Museum. *IEEE Syst. J.* **2020**, *14*, 3483–3493. [CrossRef]
4. Wang, S.-S. A BLE-Based Pedestrian Navigation System for Car Searching in Indoor Parking Garages. *Sensors* **2018**, *18*, 1442. [CrossRef] [PubMed]
5. Kriz, P.; Maly, F.; Kozel, T. Improving Indoor Localization Using Bluetooth Low Energy Beacons. *Mob. Inf. Syst.* **2016**, *2016*, 2083094. [CrossRef]
6. He, S.; Chan, S.-H.G. Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 466–490. [CrossRef]
7. Scaramuzza, D.; Fraundorfer, F. Visual Odometry [Tutorial]. *IEEE Robot. Autom. Mag.* **2011**, *18*, 80–92. [CrossRef]
8. Zhang, R.; Yang, H.; Höflinger, F.; Reindl, L.M. Adaptive Zero Velocity Update Based on Velocity Classification for Pedestrian Tracking. *IEEE Sens. J.* **2017**, *17*, 2137–2145. [CrossRef]
9. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; Massachusetts Institute of Technology: Cambridge, MA, USA, 2005.
10. Yan, H.; Herath, S.; Furukawa, Y. RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods. *arXiv* **2019**, arXiv:1905.12853 2019.
11. Yan, H.; Shan, Q.; Furukawa, Y. RIDI: Robust IMU Double Integration. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 621–636.
12. Chen, C.; Zhao, P.; Lu, C.X.; Wang, W.; Markham, A.; Trigoni, N. Oxiod: The Dataset for Deep Inertial Odometry. *arXiv* **2018**, arXiv:1809.07491 2018.
13. Hallemans, A.; Ortibus, E.; Meire, F.; Aerts, P. Low Vision Affects Dynamic Stability of Gait. *Gait Posture* **2010**, *32*, 547–551. [CrossRef]
14. Iosa, M.; Fusco, A.; Morone, G.; Paolucci, S. Effects of Visual Deprivation on Gait Dynamic Stability. *Sci. World J.* **2012**, *2012*, 974560. [CrossRef]
15. Tomomitsu, M.S.; Alonso, A.C.; Morimoto, E.; Bobbio, T.G.; Greve, J. Static and Dynamic Postural Control in Low-Vision and Normal-Vision Adults. *Clinics* **2013**, *68*, 517–521. [CrossRef]
16. Flores, G.H.; Manduchi, R. Weallwalk: An Annotated Dataset of Inertial Sensor Time Series from Blind Walkers. *ACM Trans. Access. Comput. (TACCESS)* **2018**, *11*, 1–28. [CrossRef]
17. Jacobson, W. Orientation and mobility. In *Assistive Technology for Blindness and Low Vision*; CRC Press: Boca Raton, FL, USA, 2012.
18. Flores, G.; Manduchi, R. Easy Return: An App for Indoor Backtracking Assistance. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; pp. 1–12.
19. Yoon, C.; Louie, R.; Ryan, J.; Vu, M.; Bang, H.; Derksen, W.; Ruvolo, P. Leveraging Augmented Reality to Create Apps for People with Visual Disabilities: A Case Study in Indoor Navigation. In Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility, Pittsburgh, PA, USA, 18–30 October 2019; pp. 210–221.
20. Microsoft. Path Guide: Plug-and-Play Indoor Navigation. Available online: https://www.microsoft.com/en-us/research/project/path-guide-plug-play-indoor-navigation/ (accessed on 14 November 2020).
21. Hsuan Tsai, C.; Peng, R.; Elyasi, F.; Manduchi, R. Finding Your Way Back: Comparing Path Odometry Algorithms for Assisted Return. In Proceedings of the 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops), Kassel, Germany, 22–26 March 2021.

22. Tian, Q.; Salcic, Z.; Kevin, I.; Wang, K.; Pan, Y. An Enhanced Pedestrian Dead Reckoning Approach for Pedestrian Tracking Using Smartphones. In Proceedings of the 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 7–9 April 2015; pp. 1–6.
23. Jin, Y.; Toh, H.-S.; Soh, W.-S.; Wong, W.-C. A Robust Dead-Reckoning Pedestrian Tracking System with Low Cost Sensors. In Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications (PerCom), Seattle, WA, USA, 21–25 March 2011; pp. 222–230.
24. Pai, D.; Sasi, I.; Mantripragada, P.S.; Malpani, M.; Aggarwal, N. Padati: A Robust Pedestrian Dead Reckoning System on Smartphones. In Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, 25–27 June 2012; pp. 2000–2007.
25. Zhao, H.; Zhang, L.; Qiu, S.; Wang, Z.; Yang, N.; Xu, J. Pedestrian Dead Reckoning Using Pocket-Worn Smartphone. *IEEE Access* **2019**, *7*, 91063–91073. [CrossRef]
26. Xiao, Z.; Wen, H.; Markham, A.; Trigoni, N. Robust Indoor Positioning with Lifelong Learning. *IEEE J. Sel. Areas Commun.* **2015**, *33*, 2287–2301. [CrossRef]
27. Xiao, Z.; Wen, H.; Markham, A.; Trigoni, N. Robust Pedestrian Dead Reckoning (R-PDR) for Arbitrary Mobile Device Placement. In Proceedings of the 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, Korea, 27–30 October 2014; pp. 187–196.
28. Harle, R. A Survey of Indoor Inertial Positioning Systems for Pedestrians. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1281–1293. [CrossRef]
29. Alzantot, M.; Youssef, M. UPTIME: Ubiquitous Pedestrian Tracking Using Mobile Phones. In Proceedings of the 2012 IEEE Wireless Communications and Networking Conference (WCNC), Paris, France, 1–4 April 2012; pp. 3204–3209.
30. Jayalath, S.; Abhayasinghe, N.; Murray, I. A Gyroscope Based Accurate Pedometer Algorithm. In Proceedings of the International Conference on Indoor Positioning and Indoor Navigation, Montbeliard, France, 28–31 October 2013; Volume 28, p. 31.
31. Gu, F.; Khoshelham, K.; Shang, J.; Yu, F.; Wei, Z. Robust and Accurate Smartphone-Based Step Counting for Indoor Localization. *IEEE Sens. J.* **2017**, *17*, 3453–3460. [CrossRef]
32. Edel, M.; Köppe, E. An Advanced Method for Pedestrian Dead Reckoning Using BLSTM-RNNs. In Proceedings of the 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Banff, AB, Canada, 13–16 October 2015; pp. 1–6.
33. Kang, J.; Lee, J.; Eom, D.-S. Smartphone-Based Traveled Distance Estimation Using Individual Walking Patterns for Indoor Localization. *Sensors* **2018**, *18*, 3149. [CrossRef] [PubMed]
34. Yoshida, T.; Nozaki, J.; Urano, K.; Hiroi, K.; Kaji, K.; Yonezawa, T.; Kawaguchi, N. Sampling Rate Dependency in Pedestrian Walking Speed Estimation Using DualCNN-LSTM. In Proceedings of the Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, London, UK, 9 September 2019; pp. 862–868.
35. Solin, A.; Cortes, S.; Rahtu, E.; Kannala, J. Inertial Odometry on Handheld Smartphones. In Proceedings of the 2018 21st International Conference on Information Fusion (FUSION), Cambridge, UK, 10–13 July 2018; pp. 1–5.
36. Kok, M.; Hol, J.D.; Schön, T.B. Using Inertial Sensors for Position and Orientation Estimation. *arXiv* **2017**, arXiv:1704.06053 2017.
37. Marron, J.J.; Labrador, M.A.; Menendez-Valle, A.; Fernandez-Lanvin, D.; Gonzalez-Rodriguez, M. Multi Sensor System for Pedestrian Tracking and Activity Recognition in Indoor Environments. *Int. J. Ad Hoc Ubiquitous Comput.* **2016**, *23*, 3–23. [CrossRef]
38. Flores, G.H.; Manduchi, R.; Zenteno, E.D. *Ariadne's Thread: Robust Turn Detection for Path Back-Tracing Using the IPhone*; IEEE: Corpus Christi, TX, USA, 2014.
39. Roy, N.; Wang, H.; Roy Choudhury, R. I Am a Smartphone and i Can Tell My User's Walking Direction. In Proceedings of the Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, Bretton Woods, NH, USA, 16–19 June 2014; pp. 329–342.
40. Kunze, K.; Lukowicz, P.; Partridge, K.; Begole, B. Which Way Am I Facing: Inferring Horizontal Device Orientation from an Accelerometer Signal. In Proceedings of the 2009 International Symposium on Wearable Computers, Washington, DC, USA, 4–7 September 2009; pp. 149–150.
41. Steinhoff, U.; Schiele, B. Dead Reckoning from the Pocket-an Experimental Study. In Proceedings of the 2010 IEEE international conference on pervasive computing and communications (PerCom), Mannheim, Germany, 29 March–2 April 2010; pp. 162–170.
42. Qian, J.; Ma, J.; Ying, R.; Liu, P.; Pei, L. An Improved Indoor Localization Method Using Smartphone Inertial Sensors. In Proceedings of the International Conference on Indoor Positioning and Indoor Navigation, Montbeliard, France, 28–31 October 2013; pp. 1–7.
43. Brajdic, A.; Harle, R. Walk Detection and Step Counting on Unconstrained Smartphones. In Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing, Zurich, Switzerland, 8–12 September 2013; pp. 225–234.
44. Scholkmann, F.; Boss, J.; Wolf, M. An Efficient Algorithm for Automatic Peak Detection in Noisy Periodic and Quasi-Periodic Signals. *Algorithms* **2012**, *5*, 588–603. [CrossRef]
45. Mannini, A.; Sabatini, A.M. A Hidden Markov Model-Based Technique for Gait Segmentation Using a Foot-Mounted Gyroscope. In Proceedings of the 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 30 August–3 September 2011; pp. 4369–4373.
46. Yao, Y.; Pan, L.; Fen, W.; Xu, X.; Liang, X.; Xu, X. A Robust Step Detection and Stride Length Estimation for Pedestrian Dead Reckoning Using a Smartphone. *IEEE Sens. J.* **2020**, *20*, 9685–9697. [CrossRef]

47. Ho, N.-H.; Truong, P.H.; Jeong, G.-M. Step-Detection and Adaptive Step-Length Estimation for Pedestrian Dead-Reckoning at Various Walking Speeds Using a Smartphone. *Sensors* **2016**, *16*, 1423. [CrossRef] [PubMed]
48. Sun, Y.; Wu, H.; Schiller, J. A Step Length Estimation Model for Position Tracking. In Proceedings of the 2015 International Conference on Localization and GNSS (ICL-GNSS), Gothenburg, Sweden, 22–24 June 2015; pp. 1–6.
49. Racko, J.; Brida, P.; Perttula, A.; Parviainen, J.; Collin, J. Pedestrian Dead Reckoning with Particle Filter for Handheld Smartphone. In Proceedings of the 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Alcalá de Henares, Spain, 4–7 October 2016; pp. 1–7.
50. Klepal, M.; Beauregard, S. A Novel Backtracking Particle Filter for Pattern Matching Indoor Localization. In Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-Less Environments, San Francisco, CA, USA, 19 September 2008; pp. 79–84.
51. Leppäkoski, H.; Collin, J.; Takala, J. Pedestrian Navigation Based on Inertial Sensors, Indoor Map, and WLAN Signals. *J. Signal Process. Syst.* **2013**, *71*, 287–296. [CrossRef]
52. Bao, H.; Wong, W.-C. An Indoor Dead-Reckoning Algorithm with Map Matching. In Proceedings of the 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy, 1–5 July 2013; pp. 1534–1539.
53. Chen, C.; Lu, X.; Markham, A.; Trigoni, N. Ionet: Learning to Cure the Curse of Drift in Inertial Odometry. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
54. Cortés, S.; Solin, A.; Kannala, J. Deep Learning Based Speed Estimation for Constraining Strapdown Inertial Navigation on Smartphones. In Proceedings of the 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), Aalborg, Denmark, 17–20 September 2018; pp. 1–6.
55. Feigl, T.; Kram, S.; Woller, P.; Siddiqui, R.H.; Philippsen, M.; Mutschler, C. RNN-Aided Human Velocity Estimation from a Single IMU. *Sensors* **2020**, *20*, 3656. [CrossRef] [PubMed]
56. Jamil, F.; Iqbal, N.; Ahmad, S.; Kim, D.-H. Toward Accurate Position Estimation Using Learning to Prediction Algorithm in Indoor Navigation. *Sensors* **2020**, *20*, 4410. [CrossRef]
57. Kawaguchi, N.; Nozaki, J.; Yoshida, T.; Hiroi, K.; Yonezawa, T.; Kaji, K. End-to-End Walking Speed Estimation Method for Smartphone PDR Using DualCNN-LSTM. In Proceedings of the IPIN (Short Papers/Work-in-Progress Papers), Pisa, Italy, 30 September–3 October 2019; pp. 463–470.
58. Wang, Q.; Luo, H.; Ye, L.; Men, A.; Zhao, F.; Huang, Y.; Ou, C. Personalized Stride-Length Estimation Based on Active Online Learning. *IEEE Internet Things J.* **2020**, *7*, 4885–4897. [CrossRef]
59. Klein, I.; Asraf, O. StepNet—Deep Learning Approaches for Step Length Estimation. *IEEE Access* **2020**, *8*, 85706–85713. [CrossRef]
60. Feigl, T.; Kram, S.; Woller, P.; Siddiqui, R.H.; Philippsen, M.; Mutschler, C. A Bidirectional LSTM for Estimating Dynamic Human Velocities from a Single IMU. In Proceedings of the 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Pisa, Italy, 30 September–3 October 2019; pp. 1–8.
61. Wagstaff, B.; Kelly, J. LSTM-Based Zero-Velocity Detection for Robust Inertial Navigation. In Proceedings of the 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Nantes, France, 24–27 September 2018; pp. 1–8.
62. Kayukawa, S.; Ishihara, T.; Takagi, H.; Morishima, S.; Asakawa, C. Guiding Blind Pedestrians in Public Spaces by Understanding Walking Behavior of Nearby Pedestrians. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*; Association for Computing Machinery: New York, NY, USA, 2020; Volume 4, pp. 1–22.
63. Fusco, G.; Coughlan, J.M. Indoor Localization for Visually Impaired Travelers Using Computer Vision on a Smartphone. In Proceedings of the 17th International Web for All Conference, Taipei, Taiwan, 20–21 April 2020; pp. 1–11.
64. Leung, T.-S.; Medioni, G. Visual Navigation Aid for the Blind in Dynamic Environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 565–572.
65. Ahmetovic, D.; Gleason, C.; Ruan, C.; Kitani, K.; Takagi, H.; Asakawa, C. NavCog: A Navigational Cognitive Assistant for the Blind. In Proceedings of the Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, Florence, Italy, 6–9 September 2016; pp. 90–99.
66. Riehle, T.H.; Anderson, S.M.; Lichter, P.A.; Giudice, N.A.; Sheikh, S.I.; Knuesel, R.J.; Kollmann, D.T.; Hedin, D.S. Indoor Magnetic Navigation for the Blind. In Proceedings of the 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, San Diego, CA, USA, 28 August–1 September 2012; pp. 1972–1975.
67. Fallah, N.; Apostolopoulos, I.; Bekris, K.; Folmer, E. 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, Austin, TX, USA, 5–10 May 2012; pp. 425–432.
68. Riehle, T.H.; Anderson, S.M.; Lichter, P.A.; Whalen, W.E.; Giudice, N.A. Indoor Inertial Waypoint Navigation for the Blind. In Proceedings of the 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Osaka, Japan, 3–7 July 2013; pp. 5187–5190.
69. Wang, Q.; Ye, L.; Luo, H.; Men, A.; Zhao, F.; Huang, Y. Pedestrian Stride-Length Estimation Based on LSTM and Denoising Autoencoders. *Sensors* **2019**, *19*, 840. [CrossRef]
70. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
71. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555 2014.

72. Paniit, S.M.; Zhang, W. Modeling Random Gyro Drift Rate by Data Dependent Systems. *IEEE Trans. Aerosp. Electron. Syst.* **1986**, *22*, 455–460. [CrossRef]
73. Chen, R.; Liu, J.S. Mixture Kalman Filters. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2000**, *62*, 493–508. [CrossRef]
74. Cheng, Y. Mean Shift, Mode Seeking, and Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *17*, 790–799. [CrossRef]
75. Trinh, V.; Manduchi, R. Semantic Interior Mapology: A Toolbox for Indoor Scene Description from Architectural Floor Plans. In Proceedings of the 24th International Conference on 3D Web Technology, Los Angeles, CA, USA, 26–28 July 2019; Volume 2019.
76. Horvat, M.; Ray, C.; Ramsey, V.K.; Miszko, T.; Keeney, R.; Blasch, B.B. Compensatory Analysis and Strategies for Balance in Individuals with Visual Impairments. *J. Vis. Impair. Blind.* **2003**, *97*, 695–703. [CrossRef]
77. Koskimäki, H. Avoiding Bias in Classification Accuracy-a Case Study for Activity Recognition. In Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence, Cape Town, South Africa, 7–10 December 2015; pp. 301–306.
78. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A Benchmark for the Evaluation of RGB-D SLAM Systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; pp. 573–580.
79. Seyler, S.L.; Kumar, A.; Thorpe, M.F.; Beckstein, O. Path Similarity Analysis: A Method for Quantifying Macromolecular Pathways. *PLoS Comput. Biol.* **2015**, *11*, e1004568. [CrossRef]