Finding Your Way Back: Comparing Path Odometry Algorithms for Assisted Return

Chia Hsuan Tsai
Dept. of Computer Science & Engineering
University of California Santa Cruz
ctsai24@ucsc.edu

Peng Ren
Dept. of Computer Science & Engineering
University of California Santa Cruz
pren1@ucsc.edu

Fatemeh Elyasi
Dept. of Computer Science & Engineering
University of California Santa Cruz
felyasi@ucsc.edu

Roberto Manduchi
Dept. of Computer Science & Engineering
University of California Santa Cruz
manduchi@soe.ucsc.edu

***Abstract*— We present a comparative analysis of inertial-based odometry algorithms for the purpose of assisted return. An assisted return system facilitates backtracking of a path previously taken, and can be particularly useful for blind pedestrians. We present a new algorithm for path matching, and test it in simulated assisted return tasks with data from WeAllWalk, the only existing data set with inertial data recorded from blind walkers. We consider two odometry systems, one based on deep learning (RoNIN), and the second based on robust turn detection and step counting. Our results show that the best path matching results are obtained using the turns/steps odometry system.**

Keywords **- Wayfinding, Spatial accessibility, Turn detection, Step counting, RoNIN, Dynamic programming**

I. Introduction

Wayfinding in unfamiliar places can be very challenging for people who are blind, and a number of technical solutions for supported wayfinding have been proposed in the literature over the past few years. In this contribution, we are concerned with *assisted return*, which can be seen as a particular form of wayfinding. Assisted return means providing support to a blind user who, after walking along a certain path, is trying to trace their way back to the starting point. Backtracking is a critical task that is required in multiple situations. For example, a blind person may be led by a sighted guide to a certain location for a meeting (e.g., an office in a building), only to realize that assistance is not available when needing to return to the starting point (e.g. the entrance door to the building). In other situations, a blind traveler may attempt to reach a certain location by following verbal directions. If unsuccessful, this individual would need to walk back to the starting point – potentially at the risk of getting lost in the building until sighted assistance is available. A system that could provide direction as needed to facilitate returning to the starting point may increase safety and confidence of blind pedestrians visiting new places.

It is important to note that, unlike standard wayfinding systems, assisted return *does not* require access to a map of the building being visited. This is an important advantage, given the current scarcity of open access indoor maps. In fact, mechanisms similar to assisted return have been proposed for other applications, e.g. to let users make available specific paths in a building for others to follow [1] [2]. Of course, if a map is available, the system could certainly leverage this information.

As discussed in [3], an assisted return mechanism (typically embedded in the user's smartphone) must support three tasks: (1) tracking the traveler during their "way–in" (walking from a starting point to a destination), while building a representation of the path traversed; (2) tracking the traveler during return, by matching the current sub-path with the recorded way–in path (where the way-in path is time-reversed); (3) providing directions to the user during return by means of an appropriate user interface (including providing an overall description of the path). In this work we concentrate on task (2). We consider two different path reconstruction algorithms (turns/steps representation vs. full odometry via machine learning) for tracking, and propose a new mechanism for ascertaining, at each time during return, where the user is in reference to path traversed during way-in. Due to the restrictions imposed by the ongoing COVID-19 pandemic, we could not test our system "live" with blind walkers. Instead, we devised a strategy for simulated assisted return using real data from WeAllWalk [4], the only existing set of inertial data collected from blind walkers. Specifically, we consider the data sequences for two different blind participants walking along the same route. We then progressively match the path from the first walker with the path from the second walker, pretending that the former were a time-reversed path during way-in. This simulates a situation with the same participant traversing the way-in path then returning to the starting point. We propose a metric for assessing the quality of path matching, and compare our proposed path reconstruction algorithms against this metric.

This article is organized as follows. Basic concepts and related work are presented in Sec. II. Our mechanism for path matching, is described in Sec. III. Experiments on WeAllWalk data are presented in Sec. IV. Sec. V has the conclusions.

II. Preliminaries and Related Work

Inertial-based odometry. In this work we only use data from the inertial sensors in a smartphone (iPhone 6). Inertial-based odometry [5] [6] requires no external infrastructure (such as Bluetooth Low Energy beacons [7]), and no prior calibration (e.g., Wi-Fi fingerprinting). Compared to visual-based odometry [8], inertial systems do not assume use of a camera with unoccluded visibility of the environment. The user may simply keep their smartphone in their pocket, and receive information via Bluetooth earphones or bonephones.

Pedestrian Dead Reckoning (PDR). One of the simplest odometry techniques is based on step counting while determining the user's orientation using data from the accelerometers and the gyros in the phone [9]. Note that all modern smartphone APIs produce the attitude (3-D rotation) of the phone with respect to an arbitrary but fixed reference system (as computed by sensor fusion from accelerometer and gyro data [10]). Since this data involves integration of possibly noisy and biased data, drift typically accumulates over time.

In buildings whose structure is well represented by a network of corridors, a path can normally be represented as a sequence of straight segments and turns, with discrete turning angles (typically, multiples of 90º or 45º). We will call this a *turns/steps* representation. The length of a straight segment can be expressed using units such as meters or feet, or as an approximate number of steps (where this latter unit can be preferable for some blind users.) This representation is particularly useful when describing a path verbally, which is an important functionality of a safe return system. For example, a path could be expressed as: "Walk for 50 steps, turn left, walk for 25 more steps, turn right, then your destination is 40 steps away." Other types of contextual information in terms of perceivable landmarks (e.g., "take the second corridor to the left", or "the destination is the fifth door to the right) could also be useful, but cannot be detected using inertial sensors, and would normally require access to a map. Representing paths as sequences of discrete angle turns and segment lengths simplifies, to some extent, the job of odometry computation. In practice, though, robust turn detection can be challenging. False positives could be generated, for example, when the walker stops and turns around to get their bearings. In addition, the assumption of a smartphone in a fixed orientation with respect to the walker's body fails as soon as one repositions the phone (e.g. after picking up a call). Thus, robust turn detection and path length measurements are called for even in simplified topologies such as networks of corridors at discrete angles.

We describe our algorithms for PDR elsewhere [11]. In brief, we robustly detect 45º or 90º turns using a two-stage system with a "straight-walking" detector and a Mixture Kalman Filter (MKF) for tracking orientation drift. Steps are detected using an LSTM recurrent network, similar to [12].

Learning-Based Odometry. Several odometry algorithms based on deep neural networks have been proposed in recent years. In particular, RoNIN [13] was shown to outperform comparable systems in challenging data sets. RoNIN processes inertial data using one of several possible deep network architectures, and produces motion vectors defined in reference to a fixed world frame. By integrating these motion vectors, one can easily reconstruct the path taken by the walker. A remarkable feature of RoNIN is that, by decoupling the phone's orientation from the estimated user velocity, it works seamlessly even if the phone is repositioned on the user's body while walking. We used the open source implementation of RoNIN[1].

Assisted Return. The concept of assisted return for blind walkers was introduced by Flores and Manduchi [3], who experimented with a turn/segment representation of indoor environments in a study with six blind participants. The same technology used in an assisted return system could be used to help a person follow a path previously taken by another individual. An example is given by *Clew*, an augmented reality app designed for visually impaired users [1]. Clew allows one to record a route, then load it and receive directions in accessible format when the same person (or someone else) wants to traverse the same route. Localization information is obtained via visual odometry using Apple's ARKit. A similar concept is implemented by the Path Guide Android app by Microsoft, which uses magnetic signatures and inertial data for localization [2].

WeAllWalk. WeAllWalk [4] is an annotated data set with inertial and magnetic data collected from blind walkers. Ten participants walked on a number of indoor routes. Seven participants used a long cane as a mobility tool; one used a dog guide; and two alternated use of a long cane and a dog guide. Each walker carried two iPhone 6s, and was equipped with inertial sensors tied at each ankle, which were used to produce ground truth measurements of each heel strike. The paths traversed by the walkers were divided into "segments". The time at which each walker traversed the boundary between two segments was recorded, which provides a discrete set of localization data points.

III. Path Matching

An assisted return system is designed for situations in which a blind walker has traversed a certain *way-in* route (possibly with the aid of a sighted companion), then attempts to traverse the same route in reverse (*return*). The system can provide directions such as the distance to the next turn and whether to turn left or right, and can inform the user when they are off-path. We assume that the system does not have access to a map of the place, which means that it can only operate by matching any spatial information acquired during way-in with that being acquired during return. A similar mechanism can be used to help a blind person follow a route that was traversed previously by someone else (in which case the way-in path needs not be reversed).

In theory, if odometry can be accurately recovered, one could simply match the current position estimated during return with the closest position in the way-in path. Unfortunately, large errors can be expected when relying only on inertial sensors, which calls for a more sophisticated strategy. Similarly to [14], we cast the problem as one of sub-sequence alignment, which

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

seeks the best matching of the time sequence of measurements up to the current time during return, with an initial sequence of measurements during way-in. This matching can be performed based on any available spatial (or non-spatial, e.g. magnetic signatures as in [14]) information, and can contain deletions. This problem can be expressed as a minimum cost route task over a properly defined graph, which also easily allows for enforcement of a spatial continuity prior (i.e., solutions with the user location in subsequent times jumping between two distant locations are unlikely). We formalize this concept in the following.

Let $\{t_i^{in}\}$ and $\{t_j^{ret}\}$ be the sets of time instants associated with (reversed) way-in and return, respectively. (Note that the time instants $\{t_i^{in}\}$ are actually ordered backwards with respect to the way they were collected, to account for the fact that we are matching the return path against the reversed way-in path.) For example, these sequences could correspond to the time points at which inertial data was sampled, or they could be other discrete events such as measured heel strike times. Our goal is to find an ordered, typically incomplete matching of $\{t_j^{ret}\}$ with $\{t_i^{in}\}$, such that the location of the walker at a certain return time instant is similar to the location of the walker in the associated way-in time instant. We build a graph with nodes $\{n_{i,j}\}$, where each node indicates the hypothesis that the walker at time t_i^{in} during way-in found themselves in the same position as the walker at time t_j^{ret} during return. Nodes are connected by directed edges such that node $n_{i,j}$ is connected to nodes $n_{i+1,j}$, $n_{i,j+1}$, and $n_{i+1,j+1}$. A path in the graph is thus made of nodes with monotonically non-decreasing time index in both way-in and return. Consecutive nodes in a path with a repeated index (e.g., $n_{i,j} \rightarrow n_{i+1,j}$) indicate that two time instants (in this example, during way-in) are matched to the same time instant in the other path. "Conflating" multiple time instants in one sequence allows for matching sequences with different velocities (in the example, the walker may have walked slower during way-in). We associate a cost C_{con} to edges connecting $n_{i,j}$ with $n_{i,j+1}$ or $n_{i+1,j}$. The cost of a path in this graph is the sum of node and edge costs. The cost of each node should represent the likelihood that, given the measured data, the walker was in the same location (at way-in and return) at the time instants associated with that node.

We will consider two odometry algorithms, the output of which is used in properly defined node and edge costs. The first algorithm is RoNIN [13], briefly described earlier in the learning-based category. RoNIN produces $(X(t), Y(t))$ locations, which can be used directly to define node costs as the Euclidean difference between the locations measured at t_i^{in} and at t_j^{ret}. Note that this assumes that the reference systems used to define the way-in and return path have been properly aligned. The second odometry algorithm considered is based on turns/steps representation (i.e., the path traversed is represented as a sequence of straight segments separated by turns that are multiple of 45^o or 90^o turns, along with the number of steps between turns.) We discuss our definition of node and edge costs for this representation in the following.

A. *Steps Cost*

Foot step (heel strike) information can be conveniently leveraged for path matching. For example, one could try to match paths step-by-step, allowing for a few steps to be skipped in either path to account for different stride lengths. This can be implemented by assigning appropriate node costs. Specifically, any node $n_{i,j}$ in the graph associated with a step detected in one path (i.e., a heel strike was detected during way-in at t_i^{in} or during return at t_j^{ret}) but not in the other path, is assigned a "unmatched step" cost C_{us}. This mechanism assigns low cost to paths in the graph that match way-in steps with return steps. Clearly, steps alone cannot be used for reliably path matching, as the system would simply match steps to steps in an orderly fashion, with no means to control whether matching steps were detected at similar locations. However, it can be a powerful cue complementing other measurements. Note that walkers may use different strides during way-in and return. This situation, which was observed in [3], may occur when, for example, the walker is led by a sighted guide during way-in, and needs to find their way by themselves during return. Hence, the number of steps when traversing a segment may change during way-in and return.

B. *Turns Cost*

Orientation information can be very useful for path matching. Specifically, we assume that the walker was facing a similar direction at the same location during (reversed) way-in and return. In order to measure the walker's orientation at each point, we use the two-stage turn-segment mechanism described earlier, which is well suited to corridor networks. A simple approach to enforcing consistent orientation would be to assign node costs that encourage matching equal angle turns during way-in and return. For example, similarly to the mechanism employed for step matching, any node $n_{i,j}$ associated with a turn detected in one path but not in the other, or with turns detected in both paths but by different turning angles, could be assigned a certain "unmatched turn" cost. This simple approach, though, is liable to fail in common situations with short sequences of incorrectly detected turns. For example, suppose that during way-in the walker stopped for a moment and turned their body to the left for a short time (for example to return the greetings of a passer-by) then starts to walk again. In this case, the system may incorrectly detect a left turn, followed after a short time by a right turn. During return, the walker proceeds without interruptions along the same path. Correctly matching the way-in and return paths comes at the cost of two unmatched turns. The risk is that one of the two spurious way-in turns may get (incorrectly) matched to some other distant (correct) turn during return (thus resulting in a lower unmatched turn cost), potentially creating a gross path mismatch.

This example suggests that an unmatched turn should be given a lower penalty when preceded shortly by a turn in the opposite direction, as in the example above. However, this cannot be implemented by simply assigning costs to edges or nodes in our original graph. Instead, we propose a mechanism that achieves the desired result, at the cost of increasing the number of nodes and edges in the graph. The idea is to consider the orientation of the walker at each time, something that can be obtained by accumulating detected turns. We describe an algorithm that embodies this notion in the following.

Let $\hat{I}^{in} = \{\hat{\imath}\}$ be the set of indices such that a turn by angle $\theta_{\hat{\imath}}^{in}$ was detected at time $t_{\hat{\imath}}^{in}$ during (reversed) way-in. (Similar concepts are defined for the return path, with obvious symbol modifications.) Assuming that the walker had orientation of 0° at start time, and that all turns were correctly detected, the orientation of the walker at time t_i^{in} will be:

$$O^{in}(t_i^{in}) = \left(\sum_{\hat{\imath}\in\hat{I}^{in}, \hat{\imath}\leq i} \theta_{\hat{\imath}}^{in}\right) \bmod 2\pi \qquad (1)$$

Note that, since we assumed that turn angles only take discrete values $k \cdot 2\pi/N$ (e.g., $N = 4$ or 8), the walker's orientation O takes values in the same discrete set (due to 2π periodicity). One could assign a cost to each node $n_{i,j}$ that is a function of the orientation discrepancy $O^{in}(t_i^{in}) - O^{ret}(t_j^{ret})$. However, this node cost by itself would not account for potential false detections. Suppose, for example, that we want to test the hypothesis that a turn detected at time $t_{\hat{\imath}}^{in}$ was in fact a false positive. Testing this hypothesis would require evaluating how the orientation discrepancy cost would change if this detected turn were suppressed. However, suppressing a turn affects all orientations computed for times $t_i^{in} > t_{\hat{\imath}}^{in}$. Testing the same hypothesis for all detected turns would appear to result in an exponential growth of possible user orientations at each time. Thankfully, this is not the case, due to the fact that orientations belong to a finite discrete set. We implement this concept by means of a graph augmentation strategy as discussed next.

Each node $n_{i,j}$ in our original graph is replaced with a set of N nodes $n_{i,j}^w$, where the superscript w (taking values between 0 and $N-1$) indicates the hypothesis that the orientation discrepancy between way-in time t_i^{in} and return time t_j^{ret} (that is, $O^{in}(t_i^{in}) - O^{ret}(t_j^{ret})$) is equal to $w \cdot 2\pi/N$. Nodes $n_{i,j}^w$ with $w \neq 0$ are assigned a mis-orientation cost C_{mo}. Nodes $n_{i,j}^w$ that are not associated with detected turns (i.e., $i \notin \hat{I}^{in}, j \notin \hat{I}^{ret}$) are linked with zero-cost directed edges to nodes $n_{i+1,j}^w$, $n_{i,j+1}^w$, and $n_{i+1,j+1}^w$. If a turn by angle $\theta_i^{in} = k \cdot 2\pi/N$ was detected at time t_i^{in}, node $n_{i,j}^w$ is connected to three sets of nodes:

- $n_{i+1,j}^{(w+k) mod\ N}$, $n_{i+1,j+1}^{(w+k) mod\ N}$
- $n_{i+1,j}^w$, $n_{i+1,j+1}^w$
- $n_{i,j+1}^w$

The first two connections are made under the assumption that the turn was correctly detected, which triggers an update of the way-in walker orientation (from $w \cdot 2\pi/N$ to $(w+k) \cdot 2\pi/N \bmod 2\pi$). The second set of connections represent the possibility that the turn was incorrectly detected, meaning that the way-in orientation should not be changed. The directed edges for these connections are assigned a "turn suppression" cost C_{ts}. The third connection (from $n_{i,j}^w$ to $n_{i,j+1}^w$) indicates that both time instant t_j^{ret} and t_{j+1}^{ret} are matched to t_i^{in}. The decision of whether to accept this turn is postponed till node to $n_{i+1,j}^w$ or $n_{i+1,j+1}^w$, and therefore the turn suppression cost C_{ts} is not applied here. Note that $n_{i,j}^w$ is not connected to $n_{i,j+1}^{(w+k) mod\ N}$ as this could cause an incorrect extra detected turn (e.g. in the case of a path from $n_{i,j}^w$ to $n_{i,j+1}^{(w+k) mod\ N}$ to $n_{i+1,j+1}^{(w+k+k) mod\ N}$).

Similar considerations apply for nodes associated with a turn detected during return. For nodes associated with turns detected both during way-in and return, we need to consider two additional sets of edges, representing the possibility that any such detection may or may not be correct. The cost of a path in this graph is thus a function of both the discrepancy in orientation between way-in and return, and of the number of turns that need to be suppressed to minimize this discrepancy. In the example given earlier of a sequence of two spurious turns in opposite orientations, a possible low-cost path would accept both turns, while paying a total penalty for orientation discrepancy equal to $n \cdot C_{mo}$, where n is the number of nodes in the path between the two spurious turns. Another possible path would suppress both turns, thus paying a penalty equal to $2 \cdot C_{ts}$. Both solutions are acceptable. Note that the risk of incorrectly matching one of these two spurious turns to a distant (correct) turn during return, while suppressing the other spurious turn, is small, due to the associated accumulated mis-orientation cost.

IV. Experiments

In order to test our path matching algorithm in realistic assisted return situations, we considered the inertial data collected in the WeAllWalk trials. We simulated an assisted return situation where the first walker traversed the whole path first, then the second walker traverses the same path while their path is incrementally matched with the path of the first walker. Note that each blind participant in WeAllWalk carried two iPhone 6 (although one of the phones did not work in the trial with one of the participants). Two WeAllWalk participants traversed the routes first using a long cane, then using a dog guide. We considered all pairs of traversals of each path by either two different walkers, or by the same walker using different mobility tools (cane or dog guide). In total, there were 162 such traversal pairs, on which our path matching algorithm was tested. For each of them, the optimal alignment mathing was computed incrementally for each time instant during return using the incremental Dynamic Time Warping algorithm of [14]. A windowed search space (with window size of 800) was used to reduce memory and computational cost [14].

In order to evaluate the correctness of path matching, it would be necessary to access the "ground truth" matches – i.e., the correct sequence of matching time instant pairs of $\{(t_i^{in}, t_j^{ret})\}$. WeAllWalk only records the time at which each walker transitioned between different segments ("straight" or "turn" segments). We interpolated the time matches between these discrete time/location data points by assuming that participants walked at constant speed within each segment. This gives us an approximate location of each walker at all times. Based on this information, we can compute the set of nodes $\{\hat{n}_{i,j}\}$ that represent the correct matching of t_i^{in} with t_j^{ret} (meaning that the first walker was at the same location at time t_i^{in} as the second walker at time t_j^{ret}). When evaluating the correctness of a given node $n_{l,j}$ in the graph path chosen by the path matching algorithm, we find the node $\hat{n}_{i,j}$, then record the absolute difference between t_i^{in} and t_l^{in}. This measures the error (in time instants) for node $n_{l,j}$; the overall error is the average error over the whole graph path.

TABLE 1.
PATH MATCHING ERRORS (IN SECONDS) MEASURED WITH DIFFERENT ALGORITHMS FOR OUR WEALLWALK EXPERIMENTS.

RoNIN	$k \cdot 90^o$*turns*	$k \cdot 45^o$ *turns*	$k \cdot 90^o$ *turns + steps*	$k \cdot 45^o$ *turns + steps*
8.63	5.28	6.43	**4.17**	4.50

Tab. 1 shows the recorded average errors using RoNIN vs. turns/steps representation for odometry, with turns computed either at multiple of 45° or 90°. (Note that 13% of all turns in WeAllWalk are ±45° turns.) For the turns/steps representation, we considered both the case with only turns cost activated in the graph, and with both turns and steps costs activated. It is seen from Tab. 1 that the lowest average error was obtained using the turns/steps representation, with turns at multiple of 90° and use of both steps and turns in the definition of graph costs.

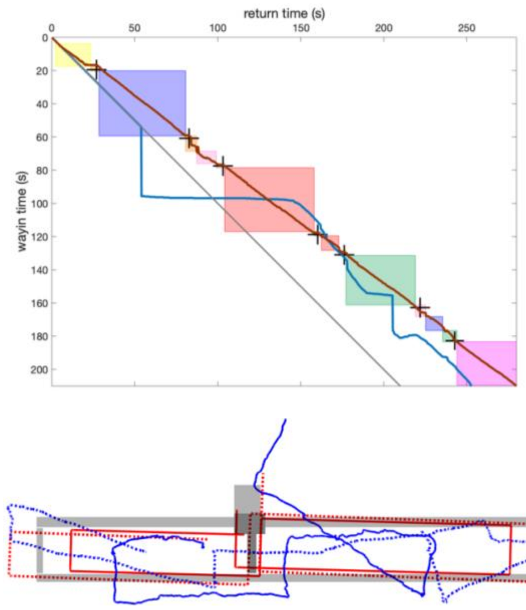

Figure 1. Top: A representation of the best matching sequence for two walkers, one using a dog guide and the other using a long cane. The colored rectangles represent the entry and exit time of each "segment" as marked in WeAllWalk. The '+' signs represent 90° turns. Blue line: RoNIN (mean error:14.9 s); Red line: $k \cdot 90^o$+steps (mean error: 0.8 s); Gray line: Baseline (mean error: 27.1 *s)*. Bottom: the reconstructed paths overimposed on the building map. Solid line: Way-in path; Dotted line: return path. Blue line RoNIN. Red line: $k \cdot 90^o$+ steps.

In order to provide some insight into the results, we show examples of path matches for pairs of walkers over the same path in Figs. 1 and 2 (top), while the reconstructed paths are shown against a map of the building (bottom). (For the $k \cdot 90^o/45^o +$ step cases, we plotted the reconstructed path assuming a stride length of 0.567 m.) In the path matching plots, the vertical and horizontal axes indicate time instants during way-in and return, respectively. The colored rectangles represent contiguous segments in the path. Specifically, the vertical coordinates of the top and bottom edge of each rectangle correspond to the times when the walker entered and exited the rectangle during way-in (as recorded in WeAllWalk), and similarly for the return path. The diagonal line joining the top left and the bottom right corners of each rectangle (not shown in the figures) represents the set of correct nodes $\{\hat{n}_{i,j}\}$ using the interpolation approximation described above. The '+' and '*' signs in the plots represent time instants (for both way-in and return) in which a 90° or 45° turn was marked in WeAllWalk. For each path matching algorithm displayed in the figures, a line is shown representing the set of nodes selected by the algorithm. Lines that are close to the diagonals of each rectangle denote satisfactory path matches. We additionally show a "baseline" match of all time instants in way-in with corresponding time instants in return $\{(t_i^{in}, t_i^{ret})\}$, which basically assumes that the two participants walked at exactly the same speed.

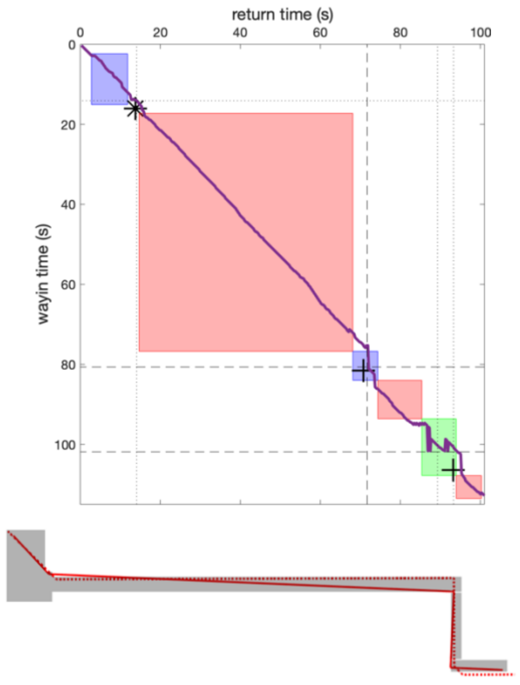

Figure 2. Top: A representation of the best matching sequence for two walkers, both using a long cane. The colored rectangles represent the entry and exit time of each "segment" as marked in WeAllWalk. The '+' signs represent 90° turns, while the '*' sign represents a 45° turn. Purple line: $k \cdot 45^o$+steps (mean error: 2.72 s). The horizontal and vertical lines show the 45° (dotted) and 90° (dashed) turns detected during way-in and return, respectively. Bottom: the reconstructed paths overimposed on the building map. Solid line: Way-in path; Dotted line: return path.

Fig. 1 compares the path matching using RoNIN (blue line) and $k \cdot 90^o +$ steps (red line) odometry. It is seen that the $k \cdot 90^o +$ steps algorithm produces a much better result than RoNIN, while both are substantially better than the baseline. Analysis of the reconstructed path (bottom plot) reveals that the paths reconstructed by RoNIN were of poor quality, due to both orientation drift and incorrect length (due to error in velocity measurement). $k \cdot 90^o +$ steps was unaffected by drift, although errors in the length of the reconstructed segments are clearly visible.

Fig. 2 shows an example using the $k \cdot 45^o +$ steps algorithm. In this case, the path had one 45^o turn at the beginning, followed by two 90^o turns. The graph shows horizontal and vertical lines, corresponding to the times when a 45^o (dotted) or 90^o (dashed) turn was detected during way-in or return, respectively. The algorithm correctly detected the 45^o turn, but the second 90^o turn was mistakenly detected as a sequence of two 45^o turns during return. The path matching algorithm was able to manage this situation correctly, although some "jitter" is noticeable (see segment marked in green), which is a consequence of the fact that dynamic programming was implemented incrementally (rather than just at the end of the return path).

V. Conclusions

We presented a new algorithm for matching the paths taken by a walker during way-in and return in the context of an assisted return system. We considered two odometry mechanisms (RoNIN and turns/segments path representations). Special provisions were taken to ensure that false turn detections were ruled out in the matching process. Our analysis showed that the $k \cdot 90^o +$ steps odometry algorithm afforded the most accurate matching results.

Our experiments were conducted on the WeAllWalk data set. We simulated a situation with a blind walker walking along a path, then back-tracking the same path, by considering two sets of recorded inertial data from different WeAllWalk participants, who traversed the same path. This allowed us to generate a large number of simulated way-in/return pairs on which to test our algorithm. It could be argued that, by comparing different walkers, this method introduces a new level of difficulty, since the gait characteristics of the two walkers on the same path may be quite different (especially when they used different mobility tools, such as one walker using a long cane and the other using a dog guide.) On the other hand, these simulations do not take into account more complex situations in which a walker may take a wrong turn during return, which needs to be detected by the system. While experiments with a real-time assisted return system (similar to [3]) will be needed, we believe that the analysis presented in this paper on WeAllWalk data is very valuable for a preliminary assessment of the proposed algorithm.

Acknowledgments

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.

References

[1] C. Yoon, R. Louie, J. Ryan, M. Vu, H. Bang, W. Derksen and P. Ruvolo, "Leveraging Augmented Reality to Create Apps for People with Visual Disabilities: A Case Study in Indoor Navigation," in *21st International ACM SIGACCESS Conference on Computers and Accessibility* , 2019.

[2] Microsoft, "Path Guide: Plug-and-play Indoor Navigation, " [Online]. Available: https://www.microsoft.com/en-us/research/project/path-guide-plug-play-indoor-navigation/.
[Accessed 7 December 2020].

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

[4] G. H. Flores and R. Manduchi, "WeAllWalk: An Annotated Data Set of Inertial Sensor Time Series from Blind Walkers," in *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2016.

[5] N. Fallah, I. Apostolopoulos, K. Bekris and E. Folmer, "The user as a sensor: Navigating users with visual impairments in indoor spaces using tactile landmarks," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[6] T. H. Riehle, S. M. Andersen, P. A. Lichter, W. E. Whalen and N. A. Giudice, "Indoor inertial waypoint navigation for the blind," in *35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013.

[7] D. Ahmetovic, C. Gleason, C. Ruan, K. M. Kitani, H. Takagi and C. Asakawa, "NavCog: a navigational cognitive assistant for the blind," in *Proc. MobileHCI*, 2016.

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

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

[10] M. Kok, J. D. Hol and T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation," *Foundations and Trends in Signal Processing,* vol. 11, no. 1-21-153, 2017.

[11] R. Peng, F. Elyasi and R. Manduchi, "Smartphone-Based Inertial Odometry for Blind Walkers," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,* Submitted.

[12] M. Edel and E. Köppe, "An advanced method for pedestrian dead reckoning using BLSTM-RNNs," in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2015.

[13] H. Yan, S. Herath and Y. Furukawa, "RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, and New Methods," arXiv preprint arXiv:1905.12853., 2019.

[14] T. H. Riehle, S. M. Anderson, P. A. Lichter, N. A. Giudice, S. I. Sheikh, R. J. Knuesel, D. T. Kollmann and D. S. Hedin, "Indoor magnetic navigation for the blind," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2012.