Indoor Trajectory Identification: Snapping with Uncertainty

Richard Wang[2], Ravi Shroff[1], Yilong Zha[1], Srinivasan Seshan[2], and Manuela Veloso[2]

Abstract— **We consider the problem of indoor human trajectory identification using odometry data from smartphone sensors. Given a segmented trajectory, a simplified map of the environment, and a set of error thresholds, we implement a map-matching algorithm in a urban setting and analyze the accuracy of the resulting path. We also discuss aggregation of user step data into a segmented trajectory. Besides providing an interesting application of learning human motion in a constrained environment, we examine how the uncertainty of the snapped trajectory varies with path length. We demonstrate that as new segments are added to a path, the number of possibilities for earlier segments is monotonically non-increasing. Applications of this work in an urban setting are discussed, as well as future plans to develop a formal theory of odometry-based map-matching.**

I. Introduction

In this paper we examine the identification of walking trajectories of people equipped with mobile phone-based odometry sensors. Specifically, we build on prior work in [1], [2] and implement a "snapping" algorithm to reconstruct human paths traversed in a real indoor environment, given an existing map of that environment. Our algorithm searches for all plausible paths within specified error bounds using the map and a segmented trajectory derived from accelerometer and gyroscope measurements.

There are three major modalities for indoor human path identification; WiFi, odometry, and vision-based. Vision-based systems rely on fixed infrastructure (cameras), odometry-based systems rely on mobile sensors, and WiFi-based systems require both fixed WiFi access points in conjunction with a WiFi-enabled mobile device [3], [4]. A significant advantage of odometry over vision or WiFi-based systems is that there is no requirement for the installation of fixed hardware (this is also an advantage over WiFi based approaches), making scaling more cost-efficient. We emphasize that the odometry-based approach does not use GPS, and in fact uses no data apart from smartphone sensor measurements and an underlying "topological map" of the space.

Our technique of map matching is borrowed from the navigation algorithms used for outdoor GPS[5], [6]. It was first used to handle indoor path identification tasks with a wheeled robot [1], [2], and proved robust in several real settings. Previous efforts relying only on cell phone odometry used probabilistic techniques like particle filters [7], which have strong independence assumptions between measurements. In contrast, our "snapping" approach focuses on using the shape of the aggregate trajectory. More recent efforts focus on leveraging unique global signatures like WiFi[8]. While these efforts focus on taking advantage of WiFi or vision sensor data, our work aims to use motion data by focusing on coarse-grain motion data instead of fine-grain motion changes that are likely due to inevitable motion while carried by a person.

In this paper, the "snapping" technique of [1] is applied to humans for the first time, and the only sensors used are the accelerometer and gyroscope found in a Samsung Galaxy S4. The process of deriving a walker's trajectory from raw data is as follows:

1) From raw odometry data, extract steps (i.e. step length and count) and heading changes.
2) Combine into a segmented trajectory.
3) "Snap" the segmented trajectory to a given map using specified error thresholds.

Given the complexity of real situations, we focus on one indoor setting, the highly structured environment of New York University's Center for Urban Science and Progress (CUSP). "Highly structured" here refers to a regime of many narrow, straight corridors and restricted spaces, as opposed to wide open, possibly curved spaces. The snapping algorithm is implemented for a complicated trajectory and fairly restrictive error thresholds, and demonstrates accurate performance. We introduce metrics to compare how uncertainty in a trajectory changes with number of path segments, compute these metrics in our example, and verify that they conform to intuition.

The work has many potential applications in urban settings. City agencies can understand how indoor public spaces are used and learn aggregate patterns of pedestrian movement (for example, to link walking patterns to health outcomes). Retailers can use trajectory knowledge to optimize store layout and cultural institutions can learn which exhibits are most viewed. We note that currently a user's path is determined by data collected from his or her personal smart phone. A thorough discussion of effective large-scale data collection strategies and solutions to anonymity and privacy concerns is outside the scope of this paper, although certainly a prerequisite to implementation. We conclude with a discussion of further research directions regarding human trajectory identification in indoor spaces.

II. Data

The snapping algorithm takes three inputs: a traversed trajectory in the form of a collection of segments, a topological map of the ambient space, and a set of error thresholds. The algorithm compares the traversed path to the topological map

[1] Center for Urban Science and Progress. New York University. 1 Metrotech Center, 19th floor, Brooklyn, NY, 11201.

[2] School of Computer Science. Carnegie Mellon University. Pittsburgh, PA. 15213

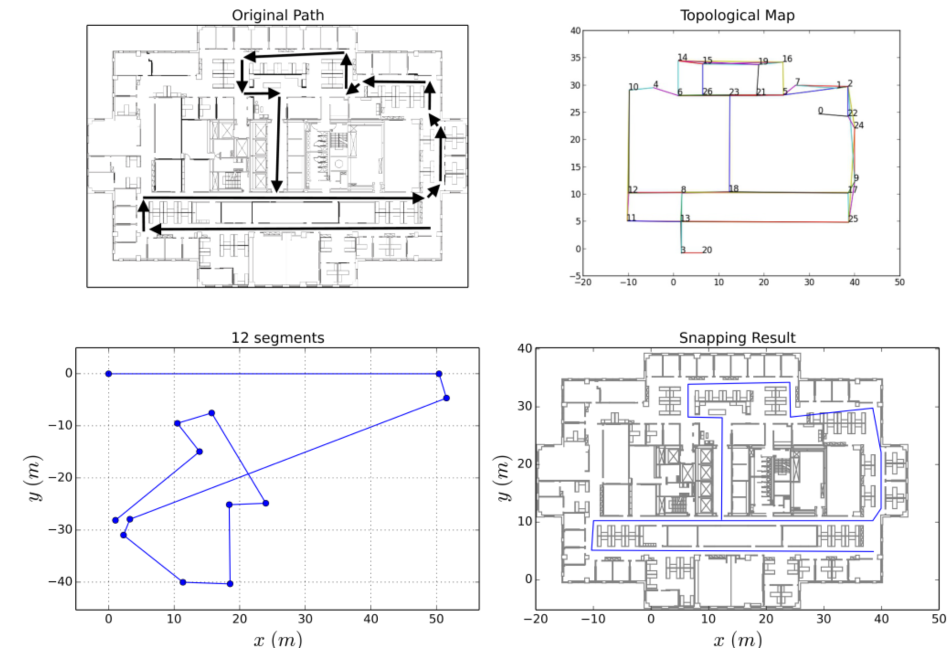

Fig. 1. The original path displays the actual path traversed at CUSP, whereas the topological map represents the intersections and hallways used by the snapping algorithm. The segments plot shows the segmented input trajectory to the snapping algorithm (note that error in both length and angle is apparent). The result of the snapping algorithm is displayed in the lower right, and conforms closely to the original path, although direction of motion is not indicated.

and determines which paths are within the error thresholds. We discuss the creation of traversed path segments from user step data in the next section.

We denote a traversed path by $\Gamma_n = \sum_{k=1}^{n}(S_k, \theta_k)$, a formal sum of segment-angle pairs. Here S_k is the k^{th} segment and θ_k is the angle between S_k and S_{k-1}, with $\theta_1 = 0$. Given Γ_n, let Γ_j, with $j \leq n$ denote the j^{th} *partial path*. We assume that each segment and angle measurement contains some unknown error from the true values, and for convenience we frequently identify a segment S with its length.

The next input to the snapping algorithm is a topological map, i.e. a simplified representation of an environment such as a floor of an office building, or platform of a subway station. A topological map is a collection of segments specified by their endpoints, representing hallways and their intersections, such that segments only intersect at endpoints. This also encodes lengths of hallways and angles between hallways. Note that a long hallway with several intersections will comprise multiple segments of varying lengths in the topological map that correspond to all possible sections of the hallway. Given a text file of the (x, y) coordinates of all walls in an environment, we use a simple point-and-click program to select intersections and construct all possible edges between intersections that do not pass through a wall.

Finally, we set two allowable error thresholds, a length threshold (dm_1, dm_2) and angular threshold da. The length threshold consists of both an allowable percentage error dm_1 and absolute error dm_2. For example, a trajectory segment S is within the length threshold of a topological map segment T, if either $\frac{|S-T|}{T} < dm_1$ or $|S - T| < dm_2$. We specify both dm_1 and dm_2 to account for large percentage errors in short segments, and large absolute errors in long segments. The angular threshold is a constant measured in radians and comparison between angles is performed similarly. Note that a complete trajectory identification implementation that converts raw data into segments and then snaps segments to a topological map may have additional parameters that affect the accuracy of the input trajectory.

III. Methods

As introduced above, the traversed path Γ_n describes the raw human trajectory in the form of segment-angle pairs. To obtain Γ_n, we first extract the steps and heading of the walker from the accelerometer and gyroscope signals of a mobile phone, in the form of length-heading pairs. The steps where a turn occurs are identified and steps between turns are integrated into longer segments, representing straight walks in each hallway. We then identify the path traversed by snapping the trajectory to a given topological map.

A. Extracting Steps and Heading

Identifying steps and heading based on sensor signals from a mobile phone is challenging. Unlike wearable sensors, the user activity, device type and position the phone is carried on the body may lead to complex variations in input data. We discuss how we are able to extract both steps and heading of the mobile device as long as the cell phone is held in a fixed orientation relative to the walker (i.e. in the hand or in a pocket).

1) Counting Steps: Instead of directly using the 3-dimensional acceleration signal, we use the magnitude of acceleration normalized to zero-mean. Thus, if $a(t) = (a_x(t), a_y(t), a_z(t))$ denotes the acceleration signal of the phone at time t, we consider the quantity $a'(t) = |a(t)| - \overline{|a(t)|}$. We denote the corresponding array of normalized acceleration measurements $\{a'(t_1), \ldots, a'(t_n)\}$ by $\mathcal{A}$.

a) Naive Threshold Crossings: The most naive approach would identify steps as zero-crossings of the normalized acceleration; unfortunately, this is unreliable in practice due to noisy sensor measurements. A slightly better approach applies a lower and upper threshold that the normalized accelerations must both cross in order to register as a step. This method is efficient and does not require advance knowledge about the period of each step; however, it assumes that the step signal is sinusoidal. It will not be able to cope with more complex accelerometer signals. For example, Figure 2a shows how the threshold crossing approach is fairly accurate when the phone is held in the hand. Each vertical line indicates the start of a new step. In contrast, Figure 2b shows how such an approach fails when the device is placed in the pocket, resulting in over-counting of steps.

b) Template Matching for Steps: With template matching, we assume a periodic function between acceleration and time but we do not manually design conditions based on prior assumptions about the shape of the acceleration signal. Instead, our template-matching algorithm automatically learns the periodic function and identifies steps based on a learned template.

There are three major steps: estimating the periodicity of steps, identifying the initial template, and then counting the number of repetitions of the template. Periodicity arises in the normalized acceleration signal as the human takes turns between stepping with their left and right foot (so each template identifies two steps). We apply a Fast Fourier Transform (FFT) to the signal to extract the period P of these two steps.

To identify an initial step template, we look at an initial time interval $I_0 = [t_0, t_0 + P]$, where t_0 is the initial time when the human started walking. Within interval I_0, we find time $t_m \in I_0$ that maximizes the acceleration signal because is usually corresponds to the person's foot hitting the ground. Roughly estimating that each step takes the same amount of time, we then choose the time interval $[t_m - \frac{P}{4}, t_m + \frac{3P}{4}]$ to represent the step template.

Finally, we count steps across the entire normalized acceleration signal by finding step template matches. For each step i, we search for the time interval $I_i = [t_i, t_i + P]$ during which the step occurred. While human steps are not perfectly synchronized, we expect that the starting time for step i will quickly follow the end of the previous step, $\{t_{i-1} + P < t_i < t_{i-1} + 5P/4\}$. We optimize for the best starting time t_i by minimizing the mean squared error of the template match. This process is repeated for each step until the entire normalized acceleration signal is processed.

As shown in Figure 2c and d, our algorithm can tolerate many different shapes of steps as long as the acceleration data is periodic. In addition, it can adjust to variations over time including slight stops and changes in pace.

2) Headings From Fixed Orientations: We also wish to extract changes in heading regardless of phone orientation. Compass readings are useful only if the relative orientation of the phone and body are known for all measurements. Additionally, compass errors may result from local magnetic fields or device misalignment. According to [9], typical mobile device compass errors, when compared to known headings, are at least 5 degrees. We therefore use both gyroscope and accelerometer data to get a raw estimate of the headings.

Assuming that the main component of acceleration is gravity (true in our context), the direction of acceleration can be used to determine where the ground is, no matter the orientation of the phone. We then approximate the angular velocity of the turning of the user's body using the acceleration-direction component of the gyroscope reading. The following equation demonstrates the relationship between ω, the estimated angular velocity of the turning of the user, and the gyroscope and accelerometer data:

$$\omega = \frac{\omega_x a_x + \omega_y a_y + \omega_z a_z}{\sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}},$$

where ω_x, ω_y and ω_z are the measures of angular velocity in the three axes of the phone.

The angular velocity ω is a good measure of the turning of the device carrier if the phone is not shaking severely relative to the carrier's body. No knowledge about how the phone is placed is required so the user may change the phone's position throughout the data collection period (e.g. taking the phone from the pocket and holding it in the hand). However, in our experiments this method will still perform poorly if the user swings their arms significantly or places the phone in a very loose pocket.

B. Segment Identification

We regard a series of consecutive steps with small total change in heading as a straight walk and a series with larger change of heading as a turn. Explicitly, consider a sequence of steps $\{x_i\}_{i=1}^m$ with respective headings $\{\psi_i\}_{i=1}^m$, let $\tau > 0$ denote the turning threshold, and w, an odd integer greater than 3, the window length parameter. For each i, consider the window W_i of w consecutive steps centered at step i (restricting the window size for i near 1 and m as necessary), and let $d\psi_i$ be the largest difference between all headings in W_i. Given a consecutive sequence $d\psi_i, \ldots, d\psi_{i+k}$, $k \geq 0$, each of which is greater than τ, we say a turn occurred at index $median(\{i, \ldots, i+k\})$. After the turns are identified, the segments are calculated as the summed length of the steps between two turns and the angles of the segments are calculated as difference in headings between consecutive turns.

Regarding the choice of parameters, w should be larger when the space is more open. For example, in a shopping mall with wide intersections, people may take more steps to finish a turn than in office environments. However, with a

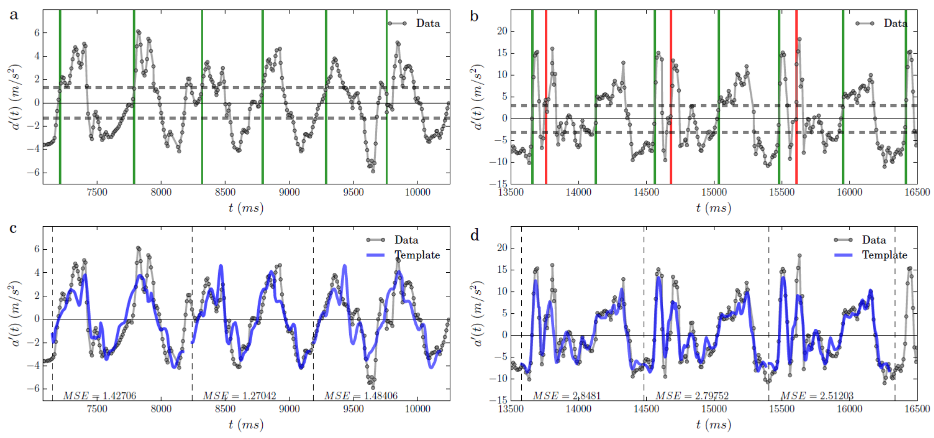

Fig. 2. Counting steps with the threshold and template approaches. In a and b, the lower and upper threshold are shown in the horizontal, dashed grey lines. The vertical lines represent the estimated start of each step, where green indicates correct registrations and red incorrect steps. In c and d the template method is used on the same signals, where the template (in blue) is matched to the signal in intervals demarcated by dashed vertical lines.

large w the window sometimes can contain more than one turn, resulting in mis-identification of turns. On the other hand, the turning threshold τ should be small enough to recognize turns, but much larger than the error in individual step headings, so avoid over-counting turns.

Note that the task of separating turns and straight walks can be ambiguous due to the difficulty in defining a turn. One can walk in a curve along a straight hallway and in a straight line along a shallow-angled intersection. This kind of ambiguity can lead to error in segment identification, but a flexible topological map can account for this. In general, we would rather consider two real segments as one rather than break one actual segment into two. For example, in Figure 1, the segment identification algorithm ignored two small turns. The modification of the topological map is to actually allow some intersections that pass through walls to be connected by artificial “hallways”. To build this topological map, one can first connect all adjacent intersections without passing through walls, and then artificially connect non-adjacent (but close) intersections which don’t require a big turn to reach one from the other.

C. Trajectory Snapping

The snapping algorithm finds all sequences of segments in the topological map that match with trajectory segments within the given error thresholds. In order to search all possibilities, the algorithm used a depth first graph search algorithm. The algorithm can be described as the recursive function **Snap**.

In function $getAdjSegments()$ the input is an intersection in the topological map and the returned value is the set of segments in the topological map with the specified intersection as an endpoint. In the function $underThreshold()$ the first input is a segment, the second input is a set of segments, and the returned value is the subset of the second input consisting of those segments within the error thresholds of the first input. Finally, the function $otherEndPoint()$ takes

Algorithm 1 **Snap** (currentPoint, Segments, path)

```
if $Segments$ is $empty$ then
    $outputpath()$;
    $exit()$;
end if
$AdjSegs = getAdjSegments(currentPoint)$;
$MatchedSegs = underThresh(Segments[0], AdjSegs)$;
for $\forall\ seg \in MatchedSegs$ do
    $newpath = [path, seg]$;
    $newPoint = otherEndPoint(seg, currentPoint)$;
    $Snap(newPoint, Segments[1:], path)$;
end for
```

two inputs, a segment and an endpoint of that segment. The returned value is the other endpoint of the segment.

When the algorithm terminates there may be several potential trajectories produced as output. We pick the one with smallest summed error as our final result. If there is no output, one can lower the thresholds $(dm_1, dm_2), da$ and rerun until a trajectory is successfully snapped. Note that there is an inverse relationship between the size of the error thresholds and the likelihood of successful snapping, and a direct relationship between the size of the error thresholds and the size of the algorithm’s search space.

IV. Results

We implement the snapping algorithm in NYU’s Center for Urban Science and Progress, which occupies the 19th floor of 1 MetroTech Center in downtown Brooklyn, NY. This setting is the floor of an indoor office building, with narrow corridors, offices, and cubicles. Most hallways are between 5 and 50 meters long, and most angles between corridors are right angles. Figure 1 shows both the underlying map of CUSP, along with the actual 12-segment long path traversed, as well as the topological map used for snapping.

Data was collected from an Android application installed

	Γ_2	Γ_3	Γ_4	Γ_5	Γ_6	Γ_7	Γ_8	Γ_9	Γ_{10}	Γ_{11}	Γ_{12}
Segment 1	2	2	1	1	1	1	1	1	1	1	1
Segment 2	2	2	1	1	1	1	1	1	1	1	1
Segment 3	-	2	1	1	1	1	1	1	1	1	1
Segment 4	-	-	1	1	1	1	1	1	1	1	1
Segment 5	-	-	-	2	2	1	1	1	1	1	1
Segment 6	-	-	-	-	3	1	1	1	1	1	1
Segment 7	-	-	-	-	-	2	1	1	1	1	1
Segment 8	-	-	-	-	-	-	1	1	1	1	1
Segment 9	-	-	-	-	-	-	-	2	2	2	2
Segment 10	-	-	-	-	-	-	-	-	2	2	2
Segment 11	-	-	-	-	-	-	-	-	-	3	2
Segment 12	-	-	-	-	-	-	-	-	-	-	1
Full path	2	2	1	2	3	2	1	2	2	3	2

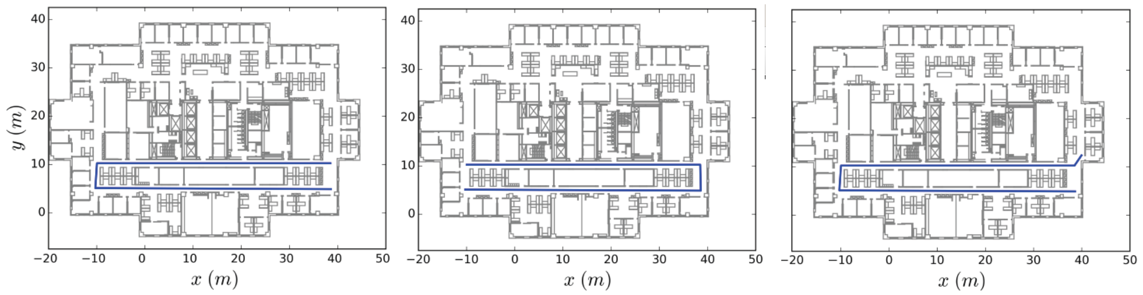

Fig. 3. The (i, j) entry in the table above shows the number of possibilities for Segment i in the partial path Γ_j, where $1 \leq i \leq 12$ and $2 \leq j \leq 12$. The j^{th} entry in the last row in the table gives the total number of possible matches for Γ_j, i.e. the maximum of all entries in the j^{th} column of the table. Note that each of the first twelve rows is monotonically non-increasing as j increases, illustrating that earlier segments are "locked in" as new segments are added to the path. The three plots give an illustration of this for Γ_3 and Γ_4. Specifically, the two left plots demonstrate the two possible matches for Γ_3, and the two possibilities for Segments 1, 2, and 3 in Γ_3. The right plot demonstrates that the addition of Segment 4 gives a unique result for Γ_4, and hence there is now only one possible match for Segments 1, 2, and 3 in Γ_4.

on the second author's handheld Samsung Galaxy S4 smartphone and converted into a segmented motion trajectory that also appears in Figure 1. We used $dm_1 = 0.25$, $dm_2 = 5$, and $da = 0.8$, roughly $\frac{\pi}{4}$ radians, as error thresholds. We took the window length parameter $w = 5$ and the turning threshold $\tau = 0.4$ radians. We also assumed a length of 0.8 meters for every two user steps when converting the raw input data into a segmented trajectory.

We have plotted the result of the snapping algorithm in the lower right panel of Figure 1, which conforms quite closely to the actual traversed path, although direction of motion is not indicated. The snapping algorithm correctly snapped all twelve segments to their best match. In this implementation there were two output paths that were admissible within the given error thresholds, but given multiple admissible paths, we rank them by total percentage error (the sum of percent angular and percent length errors over all trajectory segments) and the correct path is selected when choosing the lowest-error path.

Next, we examine how uncertainty in the snapped path varies with the addition of new segments. In particular, we look at both how the total number of possible snapped paths varies, as well as how the number of possible matches for a given segment varies.

First, we introduce some notation to facilitate a discussion of uncertainty in the snapped trajectory as the number of segments in a path increases. For a topological map M, a set of error thresholds E, and an input trajectory of n segments, $\Gamma_n = \sum_{k=1}^{n}(S_k, \theta_k)$ as above. We define

- $\rho_j(S_k)$ = the number of matches for segment S_k, considered as a segment in Γ_j, where $j \geq k$, when Γ_j is snapped to M with thresholds E.
- $\rho(\Gamma_j)$ = the number of trajectories for Γ_j that snap to M with thresholds E.

We expect $\rho_j(S_k)$ to monotonically not increase as j increases, for a fixed k. That is, adding additional constraints to Γ_j can only decrease the number of possible matches for S_k. Intuitively, segments early in a trajectory become "locked in" as newer segments are added, although the overall path may retain ambiguity. We observe this behavior also by looking at the number of matches for segment S_k, $k = 1, \ldots, 12$ in the context of the path traversed at CUSP. This is displayed in the table in Figure 3, where the number of matches for any given segment S_k only stays the same or decreases as the number of segments in the path it belongs to increases. Explicitly, S_3, for example, has two possible matches in Γ_2 and Γ_3, then has only one possible match in $\Gamma_4, \ldots, \Gamma_{12}$, as illustrated in the plots in Figure 3.

On the other hand, $\rho(\Gamma_j)$ may fluctuate non-monotonically as j increases from 1 to n. For example, $\rho(\Gamma_j) < \rho(\Gamma_{j+1})$ may occur if S_{j+1} matches several segments in the topological map within given error thresholds. On the other hand,

$\rho(\Gamma_j) > \rho(\Gamma_{j+1})$ if the addition of S_{j+1} eliminates possible matching trajectories for Γ_j. We observe this behavior in the 12-segment path traversed at CUSP, in the last row of the table in Figure 3. The total number of matches varies between 1 and 3, and increases and decreases as more segments are added.

V. Discussion

The above results provide a first application of smartphone-based human trajectory identification using the o-snap algorithm of [2] in an indoor setting. We emphasize that this method does not use any fixed infrastructure or GPS, but rather just data obtained from the sensors inside a commonly available phone. However, there are numerous further engineering challenges to be overcome, and this work may provide the foundation for myriad applications of trajectory identification in an urban setting.

First, extending the snapping approach to open spaces, rather than the narrow corridors found in office buildings. It is not clear how to best construct a topological map for an open space, such as an indoor lobby or mall plaza. One approach may create artificial hallways in a topological map of an open space, i.e. to represent the possible routes people take as a sequence of short straight lines, the algorithm search space may grow tremendously. It is also not clear that humans in a wide open space walk in straight lines; perhaps they walk along curved paths. In addition, open spaces – particularly in crowded urban settings – may also be full of people, forcing a trajectory to change direction or pause frequently. Adapting the snapping technique to take pausing and mid-segment changes in direction into account is another direction of research.

Other challenges include considerations of multiple floors and buildings. We anticipate that large-scale experiments incorporating many different users and devices would be necessary before any real-world implementation of the snapping algorithm is feasible. Furthermore, there are important privacy issues to be considered, for even if a user's identifying information is completely erased from his or her trajectory, it may still be possible to de-anonymize users by correlating trajectories with external data sets. Additionally, processes involving automatic data collection from a user's smartphone will have to be developed to encourage large-scale adoption of our technique. Derived statistics from a large corpus of trajectory data will also be a useful output for both researchers and city agencies, to understand where city residents walk and how movement patterns change over time or in response to particular events.

Finally, it is desirable to have a general theoretical framework to analyze snapping algorithm performance for a given topological map M. Suppose M is highly symmetric (for instance, the boundary of a square), then the snapping algorithm will be unable to distinguish between many different paths in the absence of a fixed starting location. Even if a starting location is given, one may have an M that is still unable to distinguish between different paths if the allowed error thresholds for displacement and heading are sufficiently, but not unreasonably, large. However, intuition suggests that for a sufficiently irregular map M, the longer a path is, the fewer potential snapped paths will be produced by the algorithm. We anticipate that such a theory may begin by creating for each M, an associated function

$$f_M : \mathbb{R}^k \times P_M \longrightarrow \mathbb{Z}_{\geq 0}, \qquad (\theta, \Gamma) \longmapsto l$$

where θ is a vector of error thresholds, P_M is the set of all possible paths on M, and l is the number of paths within the error thresholds that snap to the given path Γ on M. For θ in some range (depending on M), f_M may be a decreasing function of the length of Γ. A potential application of this theory would be to give a quantitative measure of how accurately we may expect the snapping algorithm to perform in a real-world situation.

VI. Conclusion

In this paper, we investigated the problem of snapping indoor trajectories to identify the the path traversed. We analyzed how uncertainty of the snapped trajectory varies with path length. We showed that as new segments are added, the number of possible earlier segments decreases monotonically. We also discussed the challenges of making such an approach robust in realistic urban environments.

References

[1] R. Wang, M. Veloso, and S. Seshan, "Iterative snapping of odometry trajectories for path identification," in *2013 RoboCup International Symposium*, 2013.

[2] ——, "Optimal snapping of odometry trajectories for route identification," in *Proceedings of ICRA '14, the IEEE international conference on robotics and automation*, 2014.

[3] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2000, pp. 775–784.

[4] K. Chintalapudi, A. Padmanabha Iyer, and V. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 173–184.

[5] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312–328, 2007.

[6] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation Research Part C: Emerging Technologies*, vol. 8, no. 1, pp. 91–108, 2000.

[7] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th annual international conference on mobile computing and networking*. ACM, 2012, pp. 293–304.

[8] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," in *Proceedings of the international joint conference on artificial intelligence (IJCAI)*, 2007.

[9] D. Gusenbauer, C. Isert, and J. Krösche, "Self-contained indoor positioning on off-the-shelf mobile devices," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, Sept 2010, pp. 1–9.