6-2014

O-Snap: Optimal Snapping of Odometry Trajectories for Route Identification

Richard Wang
Carnegie Mellon University

Manuela M. Veloso
Carnegie Mellon University

Srinivasan Seshan
Carnegie Mellon University

Published In
Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2014, 5824-5829.

O-Snap: Optimal Snapping of Odometry Trajectories for Route Identification

Richard Wang[1], Manuela Veloso[1], and Srinivasan Seshan[1]

Abstract— **An increasing number of wearable and mobile devices are capable of automatically sensing and recording rich information about the surrounding environment. To make use of such data, it is desirable for each data point to be matched with its corresponding spatial location. We focus on using the trajectory from a device's odometry sensors that reveal changes in motion over time. Our goal is to recover the route traversed, which we will define as a sequence of revisitable positions. Dead reckoning, which computes the device's route from its odometry trajectory, is known to suffer from significant drift over time. We aim to overcome drift errors by reshaping the odometry trajectory to fit the constraints of a given topological map and sensor noise model. Prior works use iterative search algorithms that are susceptible to local maximas [15], which means that they can be misled when faced with ambiguous decisions. In contrast, our algorithm is able to find the set of all routes within the given constraints. This also reveals if there are multiple routes that are similarly likely. We can then rank them and select the optimal route that is most likely to be the actual route. We also show that the algorithm can be extended to recover routes even in the presence of topological map errors. We evaluate our algorithm by recovering all routes traversed by a wheeled robot covering over 9 kilometers from its odometry sensor data.**

I. INTRODUCTION

The rapid proliferation of mobile and wearable devices presents incredible opportunities to capture data about environments where static devices previously could not. A lone static device equipped using sensors including temperature, Wifi, or UV can only capture a single data point at a particular location. It would require a distributed array of static sensors to reveal any spatial patterns of an environment. In contrast, a mobile device has the flexibility to capture such data and help create such spatial maps if the corresponding location for each sensor measurement is known. In addition, the growing popularity of consumer wearable devices for activity tracking could potentially benefit from route identification.

Many existing algorithms can recover the route traversed with incredible accuracy using powerful sensors including GPS, Kinect, and LIDAR. Unfortunately, only a small subset of mobile devices are equipped with such sensors. GPS performs well in outdoor environments but is insufficient for the indoor environments that our work addresses. Approaches using exteroceptive sensors including Kinect and LIDAR are wildly successful but require unwieldy form factors and usage limitations that not all devices are intended for. Wifi and Bluetooth signal strengths can also be used to localize mobile devices but require non-negligible overheads of creating detailed fingerprint maps due to the difficult to predict effects of the environment on the propagation of signal strengths.

Unlike many of these other sensors, odometry cannot uniquely distinguish a particular location from a single measurement alone; however, it does capture high resolution motion data while having low cost, small form factor, and low power consumption. The process of dead reckoning computes the device's route by recursively updating device's current position from its prior position and the current motion update unfortunately suffers significantly from cumulative drift errors.

Although the route computed using dead reckoning is poor, we recognize that odometry captures the overall shape of the route traversed by the device well. We aim to overcome drift errors by reducing the high resolution motion data into simple, large motions so that it becomes feasible to perform a brute force search for reshaping the trajectory to fit the known topological map and sensor noise model.

Our search algorithm finds the set of all feasible candidate routes from the given motion trajectory. We then rank the set of routes and select the candidate with the best opportunity of recovering the actual route. We can also inspect the candidates to see if there are ambiguous situations in which there are several good candidate routes. Our algorithm is also flexible enough to address scenarios where there may be errors in the topological maps. We demonstrate that our algorithm successfully recovers all routes traversed by a robot covering over 9 kilometres.

II. RELATED WORK

Prior work with pure dead reckoning all eventually result in significant errors over time. Careful calibration may work for short distances but will eventually require corrections [3], [4]. Particle filters have been used to correct drift errors by using walls of a known map to constrain particles to a given free space with odometry motion data [2], [5], [11], [14], [17]. As motion data cannot uniquely disambiguate the device's location, the high resolution motion updates result in large search spaces that tend to suffer in open areas. Sensors including Kinect and LIDAR are much more effective with particle filters as they can take advantage of

[1]R.Wang, M. Veloso, and S. Seshan are with the Computer Science Department, Carnegie Mellon University 5000 Forbes Ave., Pittsburgh, PA. 15213, United States {rpw,mmv,srini}@cs.cmu.edu

This research was partially supported by the Office of Naval Research under grant number N00014-09-1-1031, the National Science Foundation award number NSF IIS-1012733, and the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. The views and conclusions contained in this document are those of the authors only.

distinctive features captured from the surrounding environment to sufficiently constrain the device's possible locations.

Other works use a variety of signal sources like GPS, Wifi and Bluetooth to localize a device but these signals propagate nearly unpredictably indoors because of the complicated effects of the surrounding environment. As a result, Wifi and Bluetooth approaches for localization typically rely on known fingerprint maps of unique signal signatures, which require non-negligible overheads to create [1] [10] [7].

Topological maps are an essential piece of our algorithm so that we can focus on utilizing the shape of the overall trajectory. It has been used for both topological localization with visual features [12] for robots as well as GPS map matching algorithms [3], [6], [8], [9], [16]. The major difference is that visual features and GPS positions both capture unique signatures in an environment while our algorithm operates only on featureless, relative motion data.

We use a relative cost metric that has been shown to best take advantage of relative motion [15]. Unlike prior work, we do not use a greedy search that succumbs to local maximas. In contrast, we perform an exhaustive search that reveals the set of all reasonable routes. This allows our algorithm to be able to rank the routes and then select the optimal route. Our approach also reveals if there are multiple alternative routes to consider.

III. DEFINITIONS

We will first define the three inputs required by our algorithm: the topological map, segmented trajectory, and sensor noise model. We also briefly define dead reckoning.

A. Topological Map

Maps often use wall constraints to limit the free space of traversable areas. In contrast, a topological map is less concerned about the exact point the device may be at and more about transitions between various distinct areas of the environment. Although less accurate, such a representation has not hindered the success of GPS navigation algorithms.

A topological map is a graph with nodes and edges. If we consider an indoor office-like environment, then a node is an intersection and an edge is a hallway, where edges allow transitions between pairs of nodes. Such a representation assumes that the device commits to following a particular topological edge until an intersection point is reached. We will define a node as an intersection point (x, y) in global map coordinates. An edge is a transition between two nodes $((x_1, y_1), (x_2, y_2))$.

Figure 1 shows the difference between the walls and the topological map. One such edge is highlighted and represents a bridge connecting the two buildings. We will later remove this bridge to see if our algorithm can still recover the route traversed.

B. Segmented Trajectory

An odometry update u is the change in forward motion dm and rotation dr since the previous update. The trajectory t is then a sequence of motion updates $(u_1, u_2, ...)$ ordered by time. Odometry updates typically occur very frequently (20 Hz for our robot) and capture very small changes in motion as shown in Figure 2.

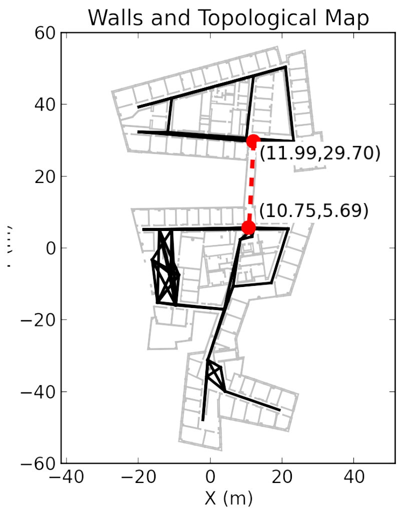


Fig. 1. The difference between wall constraints (light) and topological constraints (dark). One such topological edge (dashed) is emphasized as an example of two intersection points connected by an hallway.

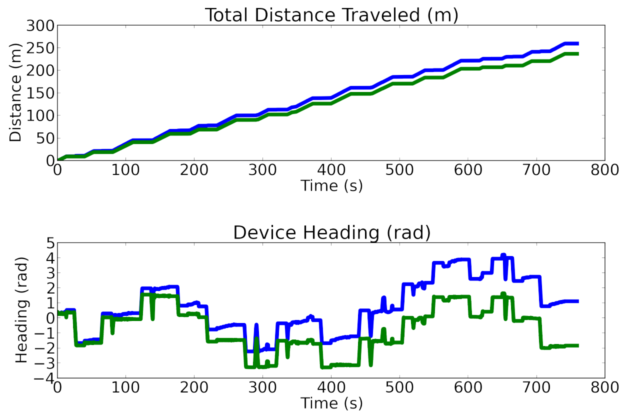
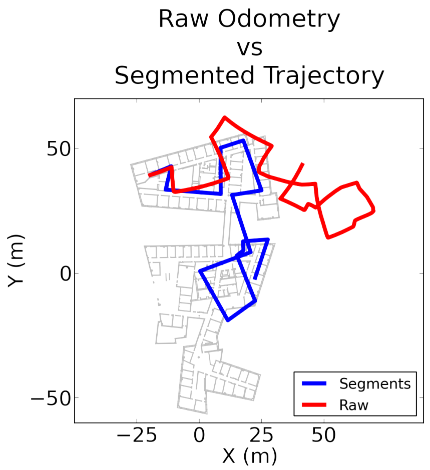


Fig. 2. Raw odometry sensor data showing the total forward distance traveled and heading changes over the route shown in Figure 3

$$u = (dm, dr)$$
$$t = u_1, u_2, ...$$

With such high resolution motion data, each update is insignificant on its own. As a result, it would be computationally infeasible to consider modifying each motion update. Instead, we segment the trajectory into a much smaller set of significant changes in motion U. This assumes that the device's motion can be characterized as a single, significant motion forward dM followed by a rotation dR. This is reasonable in office-like indoor environments where forward motions occurs while traversing a hallway followed by rotations to transition at an intersection. Figure 3 shows



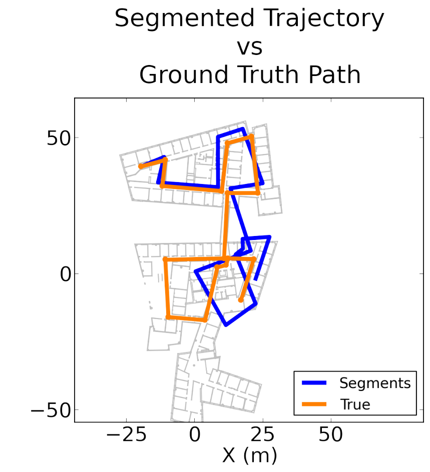


Fig. 3. Transforming the raw trajectory into simpler trajectory segments reduces drift (left). Goal is to recover the true route from the trajectory segments (right).

the segmentation of a trajectory, which appears to already significantly reduce drift. The result is a much shorter segmented trajectory T that will be computationally feasible for our algorithm to recover route candidates.

$$U = (dM, dR)$$
$$T = (U_1, U_2, ...)$$

C. Sensor Noise Model

If the ground truth route in the topological map is known, then we can compute the expected sequence of trajectory motion segments T^{expect}. The expected trajectory segments can then be directly compared to the measured trajectory segments T^{actual}.

As shown in Figure 3, although the routes look similar, the naive approach comparing their similarity using the distance between corresponding positions would not work because of drift. Our approach will separately compare the changes in motion for each corresponding intersection. This will effectively ignore drift errors from prior motion updates.

Following the relative cost metric of prior work [15], we will define the error of a single pair of motion updates to be $(\epsilon M_i, \epsilon R_i)$, which is the percentage difference of distance traveled ϵM_i and the difference in rotation ϵR_i.

$$\epsilon M_i = \frac{dM_i^{expect} - dM_i^{actual}}{dM_i^{expect}}$$
$$\epsilon R_i = R_i^{expect} - R_i^{actual}$$

We define the error over the entire route $(\eta M_{route}, \eta R_{route})$ to be maximum over the entire sequence of motion segments. As a result, the noise required to reshape a trajectory is related to the worst behaving snap and not the total sum of noise errors. This will ensure that we can compare errors across routes of different lengths.

The noise model provided to our algorithm will also be the pair $(\eta M_{model}, \eta R_{model})$. Our algorithm searches for all routes that fit within the noise model. As a result, the noise model must account for deviations from both odometry errors and the maximum deviation from the topological map. Its values must bound the worst case snapping errors.

$$\eta M = max(\epsilon M_1, \epsilon M_2, ...)$$
$$\eta R = max(\epsilon R_1, \epsilon R_2, ...)$$

D. Dead Reckoning

We will define a device's pose to be a particular position (x, y) and heading h in the coordinates of the topological map (x, y, h). Given an initial pose, dead reckoning will recursively update its pose from the arriving motion updates. Dead reckoning computes its current pose x_i, y_i, h_i using its prior pose $x_{i-1}, y_{i-1}, h_{i-1}$ and current motion update dm_i, dr_i. It computes the global displacement from the motion update and then adds it to the prior pose.

$$x_i = x_{i-1} + sin(h_{i-1} + dr_i) * dm_i$$
$$y_i = y_{i-1} + cos(h_{i-1} + dr_i) * dm_i$$
$$h_i = h_{i-1} + dr_i$$

The recursive computations reveals why errors at any step can accumulate very rapidly, resulting in significant drift errors.

IV. SNAPPING ALGORITHM

The snapping algorithm assumes that we are given the topological map, sensor noise model, and segmented motion trajectory. The algorithm will find all modifications of the segmented motion trajectory that snap to the topological map within the given sensor noise model. Our search can tolerate a more permissive noise model in exchange for additional computation.

A. Recovering Feasible Routes

The snapping algorithm finds all routes in the topological map, where the modifications required to snap all odometry trajectory segments do not violate the given sensor noise model. This can be accomplished by creating a search tree to explore all possible routes of the topological map. Each level in the tree corresponds to the corresponding index of the motion update in the trajectory segment. Although it is effectively a brute force search, the maximum depth of the search tree is bounded by the number of trajectory segment updates, which is significantly less than the number of motion updates captured. We can further prune search paths when a node cannot snap the motion update within the bounds of the noise sensor model. We also limit the search breadth of a node by limiting its children to be direct neighbors in the topological map. We have found that the search space is sufficiently small enough to be feasible in practice.

These are the rules for the search tree:

1) A node's value is the device's current pose (x, y, h).
2) A node's pose position (x, y) should correspond to an intersection in the topological map.
3) A parent node at level i can only have child nodes at level $i + 1$.
4) The position of a parent node (x_p, y_p) and child node (x_c, y_c) must have a connecting edge in the topological map.
5) A parent at level i with pose (x_p, y_p, h_p) can have a child node with pose (x_c, y_c, h_c) if applying the motion update dM_i, dR_i results in a snapping error $\epsilon M_i, \epsilon R_i$ within the bounds of the given noise model $(\eta M_{model}, \eta R_{model})$

Our current algorithm uses depth first search but other searches would also work because of the finite search space. The algorithm begins by creating a set of root nodes at level 0 initialized to one of the intersection points in the topological map. For each node, we then recursively compute the reachable intersections in the topological map from the node's current pose. A child node is created if the error in the expected and actual motion updates are within the sensor noise model $(\eta M_{model}, \eta R_{model})$. If the search reaches a depth where there are no more motion updates, then all trajectory segments have been successfully snapped. Once the search completes, we have a set of successfully snapped routes from which we can select the optimal.

Too restrictive of a noise model may not successfully recover the actual route. A more permissive sensor noise model will naturally increase the size of the search space; however, it does not reduce the quality of the routes discovered. In fact, it may discover many alternate routes, which is not necessarily undesirable because we know about the error bounds $(\eta M, \eta R)$ for each route. If there is one route that has a snapping error significantly lower than all others, then it is very likely that it is the actual route. In contrast, if several routes have similar errors, then the best route is now more ambiguous.

B. Topological Map Errors

The algorithm presented above depends on having a correct topological map. In our work, we found that intersections were much easier to mark than the hallways. As a result, we automatically created hallway connections between all intersections that do not intercept any of the wall constraints. Unfortunately, this approach may miss out on connections in topologies like ours that do not always have clear line of sight between intersections.

One approach to addressing such errors is to eliminate the constraint that a parent and its child can only have positions connected by an edge in the topological map. This would be undesirable because it would effectively ignore all edges in the topological map and result in an enormous search space. Instead, we set a user specified limit on the number of time in a route that a child that does not share a direct connection to its parent in the topological map. This simple extension allows our algorithm to be flexible enough to address a few missing connections while still taking advantage of connectivity in the topological map to find only the most likely routes and also remaining computationally feasible.

V. EVALUATION

In our evaluation, we record odometry and localization data from an autonomous, omni-directional wheeled robot [13] equipped with odometry, LIDAR, and Kinect sensors. The robot is directed to visit a sequence of waypoints throughout the environment. Each waypoint corresponds to a node on the topological map. The recorded odometry trajectory segments are given to our algorithm to see if it can recover the route.

This setup allows us to evaluate our algorithm on repeatable, real odometry sensor data for which we have the ground truth. It also introduces additional challenges for our algorithm because the robot's autonomous behavior in avoiding obstacles or wall following may violate the assumptions of direct motion between waypoints. We will show that our algorithm can still successfully recover all routes in the presence of such challenges.

A. Recovered Routes

With the given sequence of trajectory segments from the robot's odometry, our algorithm returns a set of all recovered routes. Figure 4 shows a set of routes recovered from the trajectory segments from Figure 3. Notice that the recovered routes shown looks remarkably similar except for a select few highlighted segments, which are all located in an open area where there were several possible choices. This similarity means we can be fairly confident that we have correctly identified most of the actual route. If the algorithm would have found routes with completely different start and end positions, then it would be more difficult to be confident about choosing a particular route. These types of conclusions are made possible only because our algorithm is able to provide us with the set of all candidate routes for the particular trajectory and map. Such conclusions would not

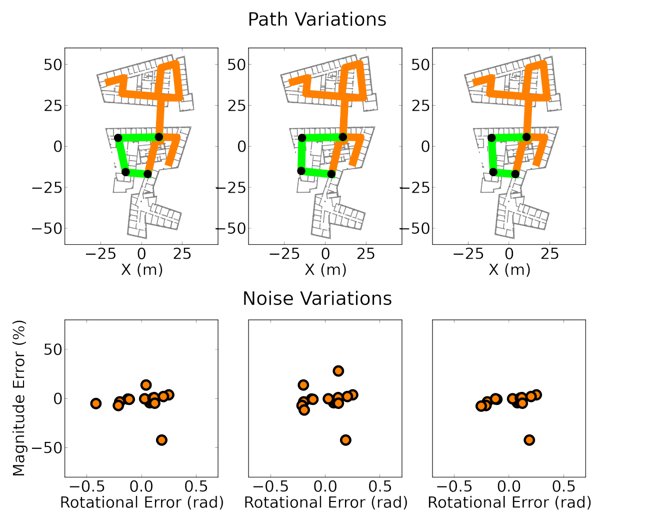


Fig. 4. Set of recovered routes that agree on the beginning and end of the route but differ in the open area (top). Corresponding snapping errors (bottom) helps to suggests best route among candidates.

be possible with other approaches including particle filters that must work with a much larger search space.

B. *Route Errors*

With the recovered routes from our algorithm, we want to be able to identify the actual route traversed. Luckily, our algorithm is also aware of the errors required to snap the trajectory for each recovered route. We can create a ranking metric to compare the routes based on these errors. Figure 4 show errors for each snapping decision from each recovered route. In general, we would expect that the actual route would require the most conservative noise model. A visual inspection of the choices in Figure 4 would correctly identify the route on the right as the best candidate for the actual route. Snapping errors uncovered by our algorithm will be essential in helping us to distinguish and rank the recovered routes.

C. *Optimal Routes*

The robot was directed to follow the sequence of way-points specified by the actual routes shown in Figure 5. Each route was repeated for five iterations to evaluate repeatability of our algorithm. Without any corrections, the recorded trajectory segments result in fairly poor routes with many being completely outside the map. Drift errors for the routes differ depending on a combination of difficulty and length of the route, directions of turns, and other autonomous behaviors.

Each route was repeated for five iterations. The algorithm was given a very permissive noise model of $(eM, eR) = (65\%, .8radians)$. From the set of recovered routes, we rank them based on a simple sum of squared distance on the maximum snapping error of each route $\sqrt{eM^2 + eR^2}$. Figure 6 shows the ranking of the actual route to the total number of recovered routes. In most cases, the actual route has the highest rank. As shown in Figure 4, while the

Iter	Route				
	1	2	3	4	5
1	**2** (6)	**3** (6)	**1** (24)	**1** (9)	**1** (96)
2	**1** (6)	**3** (6)	**2** (24)	**1** (18)	**1** (96)
3	**1** (6)	**3** (6)	**1** (24)	**1** (9)	**1** (96)
4	**1** (6)	**3** (6)	**1** (24)	**1** (18)	**1** (96)
5	**1** (6)	**3** (6)	**2** (24)	**4** (18)	**1** (96)

Fig. 6. Ranking of the recovered actual route to the total number of routes. Ranking is computed with a sum of squared difference of snapping errors.

Iter	Route				
	1	2	3	4	5
1	**6** (644)	**1** (96)	**17** (480)	**7** (27)	**17** (5184)
2	**1** (12)	**1** (96)	**17** (480)	**3** (54)	**17** (5184)
3	**6** (760)	**1** (96)	**17** (480)	**7** (2880)	**217** (5184)
4	**7** (760)	**1** (96)	**21** (576)	**15** (5307)	**97** (5184)
5	**5** (638)	**1** (48)	**17** (480)	**19** (5067)	**95** (5184)

Fig. 7. Ranking of the recovered actual route to the total number of routes with introduced topological map errors highlighted in Figure 1.

number of alternative routes appears large, many of them are small variations of the overall actual route. Nevertheless, the success of our simple ranking algorithm affirms the observation that the actual route will typically have the most conservative snapping error bounds.

D. *Topological Map Errors*

To evaluate if our algorithm can address topological map errors, we removed the bridge highlighted in Figure 1, which is essential for all of the routes. Without this bridge, our original algorithm failed to identify any routes because it could not proceed once it reached the bridge. We performed the search again by expanded the search to allow for two map errors. The extended algorithm was able to recover a set of routes that includes the actual route. Unfortunately, the increased search space results in a significantly increased number of route candidates even though most are minor variations for traversing open areas. Although not as reliable as before, we see that the simple ranking function tends to place the actual route in the top 5% of potential candidates.

VI. CONCLUSION AND FUTURE WORK

Given trajectory segments from odometry, we have shown that our algorithm can successfully find the entire set of candidate routes even in the presence of topological map errors. We are most interested in recovering the actual route and we have shown that a simple ranking function can help to differentiate the routes. Opportunities remain to explore more robust ranking functions, better compress the routes generated to eliminate minor route variations, and reduce the exponential growth of routes in the presence of map errors.

REFERENCES

[1] Paramvir Bahl and Venkata 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*, volume 2, pages 775–784. IEEE, 2000.

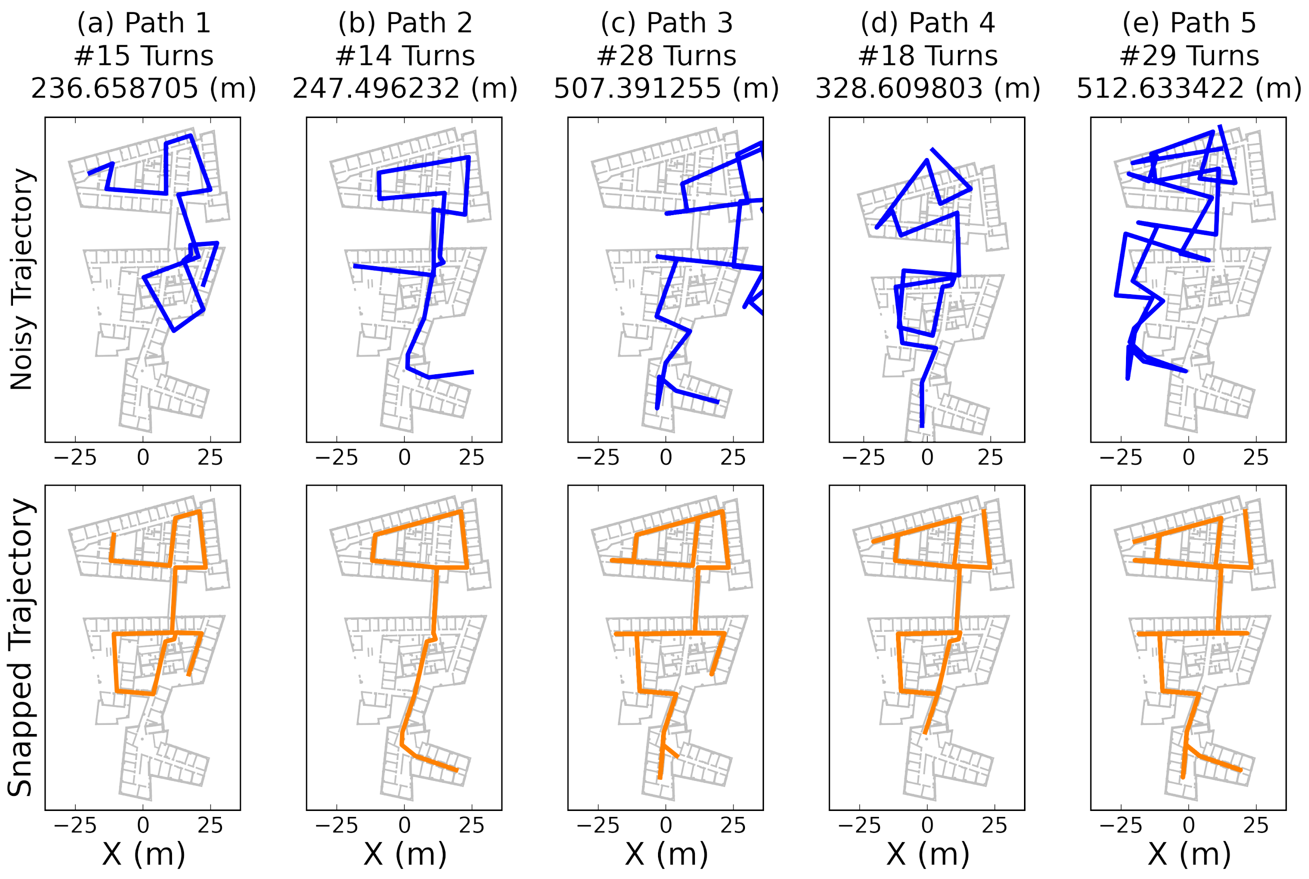


Fig. 5. The five routes traversed by the robot with trajectory segments recorded from odometry (top) and the actual routes (below).

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

[3] Nakju Doh, Howie Choset, and Wan Kyun Chung. Accurate relative localization using odometry. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1606–1612. IEEE, 2003.

[4] Alonzo Kelly. Fast and easy systematic and stochastic odometry calibration. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3188–3194. IEEE, 2004.

[5] Masakatsu Kourogi, Tomoya Ishikawa, Yoshinari Kameda, Jun Ishikawa, Kyota Aoki, and Takeshi Kurata. Pedestrian dead reckoning and its applications. In *Proceedings of Lets Go Out Workshop in conjunction with ISMAR*, volume 9, 2009.

[6] Damien Kubrak, C Macabiau, M Monnerat, and ML Bouchert. Vehicular navigation using a tight integration of aided-gps and low-cost mems sensors. In *Proceedings of the ION NTM*, 2006.

[7] Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. Push the limit of wifi based localization for smartphones. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 305–316. ACM, 2012.

[8] Washington Y Ochieng, M Quddus, and Robert B Noland. Map-matching in complex urban road networks. *Revista Brasileira de Cartografia*, 2(55), 2009.

[9] Mohammed A Quddus, Washington Y Ochieng, and Robert B Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[10] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304. ACM, 2012.

[11] Alberto Serra, Davide Carboni, and Valentina Marotto. Indoor pedestrian navigation system using a modern smartphone. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, MobileHCI*, volume 10, pages 397–398. Citeseer, 2010.

[12] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1023–1029. IEEE, 2000.

[13] Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Susana Brandao, Tekin Mericli, and Rodrigo Ventura. Symbiotic-autonomous service robots for userrequested tasks in a multi-floor building. *Under submission*, 2012.

[14] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210. ACM, 2012.

[15] Richard Wang, Manuela Veloso, and Srinivasan Seshan. Iterative snapping of odometry trajectories for path identification. *2013 RoboCup International Symposium*, 2013.

[16] Christopher E White, David Bernstein, and Alain L Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, 8(1):91–108, 2000.

[17] Oliver Woodman and Robert Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 114–123. ACM, 2008.