Iterative Snapping of Odometry Trajectories for Path Identification

Richard Wang, Manuela Veloso, Srinivasan Seshan

Computer Science Department
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA, USA
{rpw,mmv,srini}@cs.cmu.edu

Abstract. An increasing number of mobile devices are capable of automatically sensing and recording rich information about the surrounding environment. Spatial locations of such data can help to better learn about the environment. In this work, we address the problem of identifying the locations visited by a mobile device as it moves within an indoor environment. We focus on devices equipped with odometry sensors that capture changes in motion. Odometry suffers from cumulative errors of dead reckoning but it captures the relative shape of the traversed path well. Our approach will correct such errors by matching the shape of the trajectory from odometry to traversable paths of a known map. Our algorithm is inspired by prior vehicular GPS map matching techniques that snap global GPS measurements to known roads. We similarly wish to snap the trajectory from odometry to known hallways. Several modifications are required to ensure these techniques are robust when given relative measurements from odometry. If we assume an office-like environment with only straight hallways, then a significant rotation indicates a transition to another hallway. As a result, we partition the trajectory into line segments based on significant turns. Each trajectory segment is snapped to a corresponding hallway that best maintains the shape of the original trajectory. These snapping decisions are made based on the similarity of the two curves as well as the rotation to transition between hallways. We will show robustness under different types of noise in complex environments and the ability to propose coarse sensor noise errors.

1 INTRODUCTION

Sensor-equipped mobile devices can sense and record information about their surrounding environments. Mobile devices are presented with incredible opportunities to collect data where static devices cannot. For example, a stationary thermometer is rather uninteresting because it only measures temperature at a single location. In contrast, a mobile device equipped with a thermostat could create a temperature map to reveal patterns and perhaps even identify drafty windows. Interesting spatial maps could be created from many other sensors including UV, air quality, radiation, and Wifi. Devices that move can help reveal rich information about our surrounding environments. One can create these spatial maps if the spatial locations for each corresponding sensor measurement are known. If we knew the path traversed by the device, then we can estimate such spatial locations. In this work, we will address the offline problem of estimating the path traversed by a device in an indoor environment from odometry and a known map.

There exist many techniques to identify the path traversed from advanced sensors like Kinect, LIDAR, Sonar and GPS. However, these sensors are typically limited to a subset of mobile devices like cars and robots. In addition, extrinsic sensors are challenged by scenarios in which there are varying light conditions and dynamic objects and GPS is challenged by indoor environments. Odometry is advantageous because it captures high-resolution changes in motion while also being low cost, occupying a small form factor, and consuming little power. This makes it accessible to a large number of very mobile devices including pedometers, health monitors, cell phones, and blind robots. Odometry faces unique challenges because its measurements are relative. While each measurement alone is a relatively insignificant change in motion, aggregating a lengthy sequence of odometry measurements can reveal a unique shape traversed by the device. In contrast, a single GPS measurement can reveal the global position of the device.

The process of computing the path traversed by odometry is called dead reckoning. Errors in odometry accumulate and typically result in wildly incorrect position estimates. While not ideal for computing the exact path traversed, odometry is very good at capturing the relative shape of the path traversed. Figure 1 is an example of odometry collected from wheel encoders of an omnidirectional robot. Most of the cumulative errors are a result of drift errors that occurred when the robot turned at intersections. With the map, it becomes obvious that the path from odometry does not travel on any traversable paths. Our approach will use the given map in order to correct these significant cumulative errors by matching the shape of the trajectory to traversable paths.

We address offline path identification because the position at a particular point is unclear until one considers the subsequent trajectory that follows. For example, one cannot disambiguate two right turns in the same hallway until the subsequent trajectory reveals a shape that is unique to only one of the two choices. To utilize the shape of the trajectory, we borrow techniques from vehicular GPS map matching. GPS does not consider the infinite space of exact positions of the vehicle on a road. Its primary concern is to match the vehicle to the correct road segment. GPS measurements are snapped to roads by using one of several geometric map matching techniques. Most relevant to our approach is curve-to-curve matching because it computes the distance between an entire sequence of GPS measurements to that of an entire road segment. One can add topological map matching to exploit the connectivity constraints when transitioning between different road segments. Instead of considering all nearby roads for snapping, topological map matching only snaps to road segments connected to the road that the vehicle is currently matched to. The combination of these

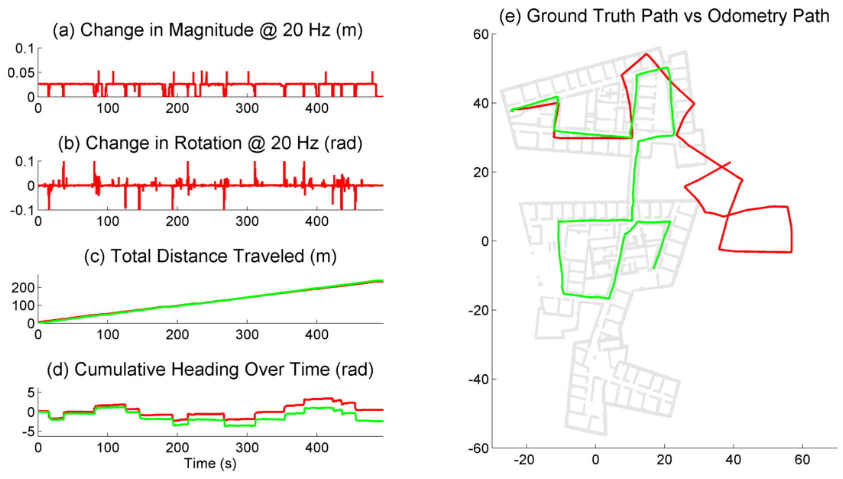

Fig. 1. Wheel encoder odometry (red) collected from a particular traversal of an indoor environment compared to ground truth collected by localization with a Kinect (green). The (a) changes in forward motion are (c) accumulated over time to reveal that the device has traveled over 200 m. The (b) changes in rotation are accumulated to reflect its (d) cumulative heading. At the end of the path, the distance traveled deviated from ground truth by only 6 m while there is much more significant drift in the device's heading. This is (e) visualized by comparing the actual truth path to the path constructed from dead reckoning.

techniques leads to very robust navigation algorithms for GPS [7] in real world scenarios.

Our approach has many parallels to these GPS map matching algorithms. Instead of snapping to roads, we will be snapping to hallways. In this work, we will assume indoor, office-like environments with straight hallways. To evaluate how well a trajectory segment matches a hallway segment, we perform curve-to-curve matching by computing their similarity instead of distance. Distance is ideal for GPS because better matching roads will be closer to the global position measurements. A similarity metric is better suited for comparing the relative shapes of two curves, which is the case with odometry. We use topological map matching to both enforce connectivity between hallways and also to compare the change in heading required for the device to transition to another hallway. A good candidate hallway will require a change in heading that matches very well to the change in rotation measured by odometry. The combination of these two relative metrics makes our approach more robust in identifying paths that better match the shape of the trajectory from odometry.

We will discuss related work in Section 2, introduce relevant definitions in Section 3, formalize our snapping algorithm in Section 4, and evaluate our algorithm in Section 5.

2 RELATED WORK

Prior works have attempted to improve odometry with careful calibration [1] [4]. While one can localize over short distances [2], odometry will eventually succumb to dead reckoning errors and require corrections to fix these cumulative errors.

Correcting dead reckoning errors has been addressed in prior work with cell phones by using walls of a known map [5] [9] [12] [8]. Particle filters were used to sample the space of reachable paths. Odometry measurements were used to update the motion of the particles. Uncertainty is added to these motion updates because odometry is noisy. The walls of a known map help to constrain uncertainty of the particles because the device should not be able to pass through a wall. The problem addressed is similar to our work but our approach does not attempt to recover the exact positions traversed. As a result, we can focus on high level decisions at hallway intersections to ensure more robust global paths identified and not require being given an accurate sensor model.

Other work attempts to automatically collect unique signatures in an environment from various sensors. These signatures include Wifi [3] [6] as well as a combination of different sensors including magnetic signatures [10]. Odometry is unique because it can capture high-resolution changes in motion with a level of detail that unique signatures cannot capture. There are opportunities in the future to complement position estimates from these unique signatures with the motion trajectory from odometry.

Our algorithm borrows many techniques from GPS map matching [7] [11]. We focus on taking advantage of basic geometric and topological map matching techniques because odometry excels at capturing the relative shape of the traversed path. While distance metrics are ideal for global GPS measurements, some modifications are required so that these map matching techniques are robust for relative sensors. Our approach focuses on the shape of the trajectory by measuring similarity to make better snapping decisions for odometry.

3 DEFINITIONS

We consider two dimensional path identification in a Cartesian map. Odometry sensor data is assumed to be captured on the same plane. We consider devices traveling directly towards their destination. We require a few definitions:

Odometry update u is composed of a change in forward motion and rotation since the last update. A trajectory T is a sequence of odometry updates. A pose p is a position and direction on the given map. A path P is a sequence of poses.

$$\begin{aligned} &u = \{dm, dr\} \\ &T = \{u_1, u_2, u_3, ...\} \\ &p = \{x, y, a\} \\ &P = \{p_1, p_2, p_3, ...\} \end{aligned}$$

Computing the path of a trajectory, also called dead reckoning, can be computed recursively given an initial pose p_0. A pose p_i is computed from the pose of the previous step p_{i-1} and odometry update u_i.

$$\begin{aligned}
&DeadReckoning(p_{i-1}, u_i) \rightarrow p_i : \\
&\qquad p_i^a = u_i^{dr} + p_{i-1}^a \\
&\qquad p_i^y = p_{i-1}^y + sin(p_i^a) * u_i^{dm} \\
&\qquad p_i^x = p_{i-1}^x + cos(p_i^a) * u_i^{dm}
\end{aligned}$$

Odometry updates arriving at approximately 20 Hz tend to be very small changes in motion. As a result, we will aggregate sequences of odometry updates to form trajectory segments. We consider environments with only straight hallways. Trajectory segments can then be partitioned by identifying significant rotations. A trajectory segment approximates a sequence of odometry updates as a single, large change in motion dM and rotation dR. This will result in a straight trajectory segment that ignores the minor rotations of the aggregated odometry updates. A trajectory T is now a sequence of trajectory segments s. The dead reckoning computation remains the same except the updates are now over these more significant trajectory segments s.

$$\begin{aligned}
&s = \{u_i, u_{i+1}, u_{i+2}, ...\} \approx \{dM, dR\} \\
&T = \{s_1, s_2, s_3, ...\}
\end{aligned}$$

$$\begin{aligned}
&DeadReckoning(p_{i-1}, s_i) \rightarrow p_i : \\
&\qquad p_i^a = s_i^{dR} + p_{i-1}^a \\
&\qquad p_i^y = p_{i-1}^y + sin(p_i^a) * s_i^{dM} \\
&\qquad p_i^x = p_{i-1}^x + cos(p_i^a) * s_i^{dM}
\end{aligned}$$

4 SNAPPING

Our algorithm snaps trajectory segments to their corresponding hallways. We will segment the trajectory so that each trajectory segment corresponds to an entire hallway segment. If we assume an environment with only straight hallways, then the trajectory segment should be partitioned into straight segments. We will first explain how the hallways and trajectories are segmented, then how these are processed by the snapping algorithm, and then the relative metrics to best use trajectories from odometry.

4.1 MARKING HALLWAY SEGMENTS

Our assumption of straight hallway means that we do not need to consider curved hallways. Examples of the types of indoor environments that we consider are shown in Figure 2. If hallway intersection points are known, then hallways can be identified by connecting all hallway intersection points that do not violate a map constraint. Automatic identification of intersection points is difficult because they need to be carefully placed to have high visibility with surrounding adjacent hallways. This is a challenge especially for open areas. As a result, we manually mark the hallway intersection points as a one-time process for a given map.

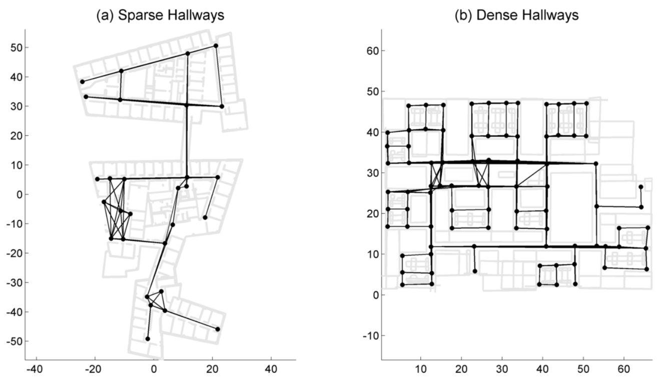

Fig. 2. Indoor environments following our assumption of straight hallways marked by the darker lines.

4.2 EXTRACTING TRAJECTORY SEGMENTS

Odometry sensors capturing measurements at 20 Hz capture very small changes in motion that individually are quite insignificant. We want to aggregate sequences of odometry updates into trajectory segments $s = \{u_i, u_{i+1}, u_{i+2}, ...\}$ to match with hallway segments. Given that the environment is composed of only straight hallway segments, we can represent each trajectory segment as a large odometry update $s \approx \{dM, dR\}$. Odometry updates are partitioned based on identification of large turns that should indicate when the device is transitioning to another hallway.

Significant rotation peaks are identified by applying a sliding window of 30 frames over the changes in rotation from odometry. This will identify the peak rotations as seen in Figure 1(b) that correspond to significant heading changes in Figure 1(d). The trajectory will be segmented based on these peak rotations. dR is assigned the value of the windowed peak rotation. dM is the total change in motion forward over the sequence of odometry updates occuring since the last significant peak rotation. We have found that this simple approach is sufficient for odometry captured by wheel encoders of an omni-directional robot.

4.3 SNAPPING TRAJECTORIES

Our snapping algorithm iteratively snaps each trajectory to the best matching hallway. As shown in Algorithm 1, the device's poses are recorded at each hallway intersection and appended to form the hallways intersections traversed. These poses can then be connected by their corresponding trajectory segments to form the path traversed. This reflects our algorithm's focus on making robust, high-level decisions at hallway intersection points as opposed to attempting to estimate the exact poses of the device. It also means that our algorithm is efficient because it does not need to consider modifications to the intermediate odometry measurements that form the trajectory segment.

We use both geometric and topological map matching from vehicular GPS algorithms. The function *GetAdjacentHalls* uses topological map matching by limiting the candidate hallways for the device to transition to. The function *GetBestMatch* uses geometric map matching to compute the similarity of the trajectory update to the candidate hallways. The combination of these two techniques means that our algorithm will prefer identifying paths that have a similar shape with the noisy trajectory. This is a benefit because odometry is good at capturing the relative shape of the path traversed.

Algorithm 1 Trajectory Snapping

```
1:  function TrajectorySnap(Hallways, TrajSegments, PoseInit)
2:      $PoseCurr \leftarrow PoseInit$;
3:      $IntersectionsTraversed \leftarrow [PoseCurrent]$;
4:
5:      for $TrajUpdate$ in $TrajSegments$ do
6:          $Candidates \leftarrow$ GetAdjacentHalls($Hallways$, $PoseCurr$)
7:          $SnappedHallway \leftarrow$ GetBestMatch($Candidates$,$PoseCurr$,$TrajUpdate$)
8:          $PoseNext \leftarrow$ GetNextIntersection($SnappedHallway$,$PoseCurr$)
9:
10:         $IntersectionsTraversed \leftarrow [IntersectionsTraversed; PoseNext]$;
11:         $PoseCurr \leftarrow PoseNext)$
```

4.4 DECIDING BETWEEN CANDIDATE HALLWAYS

Our algorithm should prefer hallway candidates that best follow the relative shape of the noisy trajectory. Each trajectory segment is a rotation and forward motion. Applying this trajectory update to the device's current pose should move it from the current hallway intersection to the next hallway intersection. However, there can be many hallways to choose from. We need a metric that will prefer candidate hallways that best matches the measured trajectory update.

Distance is ideal for GPS because global measurements are closer to roads that are better candidates. With relative measurements, drift can result in significant distances between the best hallway candidate and the trajectory. Our relative metric considers the current pose of the device and computes the relative change in motion to transition to each of the candidate hallways. The hallway that best matches the trajectory update will be chosen for snapping. Currently, we use a simple similarity function that computes the square root of the sum of squared differences between the percentage difference in forward motion and the difference in angular rotation. While future work can better engineer these error components, it is most important that the relative metric measures these components separately and then combines them.

As we can see in Figure 3(a), the distance metric would incorrectly snap to the shorter hallway candidate (0,5). In contrast, the relative metric Figure 3(b) would recognize that the change in rotation is equivalent for both candidates but the magnitude of the trajectory matches the longer hallway candidate (0,10) much better. There are many other situations where distance will making puzzling decisions while our relative metric snaps to more intuitive hallways. Because relative snapping cannot recover once a poor decision is made, improving robustness is very important for the success of our algorithm.

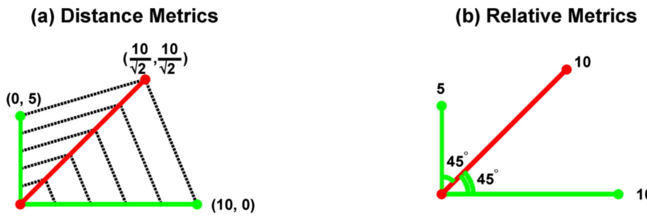

Fig. 3. Deciding which candidate hallway (green) to snap the trajectory update (red). A distance metric would select the shorter hallway because the sum of sampled distances (dashed) is smaller. The relative metric separates rotation and motion errors and selects the longer hallway because it only differs from the trajectory update by rotational error while the shorter hallway differs by both rotational and motion errors.

Algorithm 2 Best Matching Hallway

```
1: function GetBestMatch(Candidates, PoseCurr, TrajUpdate)
2:     $BestSnapChoice \leftarrow \emptyset$
3:     for $Hallway$ in $Candidates$ do
4:         $hM \leftarrow$ ComputeRelativeMotion($Hallway$, $PoseCurr$)
5:         $hR \leftarrow$ ComputeRelativeRotation($Hallway$, $PoseCurr$)
6:         $DiffMotion \leftarrow TrajUpdate.dM - hM$
7:         $DiffRotation \leftarrow TrajUpdate.dR - hR$
8:         $PercentageDiffMotion \leftarrow DiffMotion/hM$
9:
10:        $Similarity \leftarrow \sqrt{PercentageDiffMotion^2 + DiffRotation^2}$
11:        if isMoreSimilar($Similarity$, $BestSnapChoice$) then
12:            $BestSnapChoice \leftarrow Hallway$
```

4.5 ESTIMATING COARSE ODOMETRY NOISE

An important property of our algorithm is that it does not require a given sensor noise model. This is possible because we do not attempt to consider the infinite space of exact positions traversed by the device. Instead, we only consider a finite number of hallway segments and force the trajectory to fit to these hallways. As a result, the relative metrics we use only need to select the best choice among the finite hallway candidates. This allows our algorithm to compute the modifications required to snap each trajectory update to its corresponding hallway. These modifications can be interpreted as the coarse odometry noise estimates of our algorithm. We can then use these noise estimates to recognize when a potentially unsuccessful snap has occurred by looking for modifications that are outliers of normal behavior.

5 RESULTS

We wish to show that our algorithm is robust and operates in real-world scenarios. We will use an extensive simulation in a challenging office environment to evaluate robustness. We will then show that our algorithm can correct errors in wheel encoder odometry from an omni-direction robot and successfully proposes coarse sensor noise errors.

5.1 Evaluating Robustness

We took the map of a real indoor office environment as shown in Figure 2(b) and performed simulations to evaluate the robustness of our snapping algorithm. This environment is especially challenging because there are many similar turns that our algorithm can incorrectly snap to. In our simulation, we take a single ground truth path, extract its trajectory segments, and then add random noise to evaluate our algorithm. Uniform noise is added by taking each trajectory segment $\{dM, dR\}$, stretching the forward motion of the device by a percentage error $\eta = uniform(-\alpha, \alpha)$, and adding rotational noise $\epsilon = uniform(-\beta, \beta)$. This results in a noisy trajectory segment $\{dM(1 + \eta), dR + \epsilon\}$.

We want to show how the success rate of snapping evolves as more noise is injected into the simulation. In our simulation, we perform 100 iterations of each combination of noise. We can see in Figure 4(a-d) the types of paths that our algorithm can successfully identify the correct path traversed. We can see that our algorithm can robustly handle fairly significant noise from the gradient of success rates in Figure 4(e). Our snapping algorithm fails when it makes an incorrect snapping decision. This results from our algorithm making greedy decisions from local information and potentially can discard the globally optimal solution. Instead of maintaining a single guess, we could maintain a set of unique guesses. Figure 4(f) shows the increased success rate when our algorithm is augmented with a set of 5 guesses. When incorrect decisions are made, it is more likely that another guess will have made the correct decision. As we increase the number of unique guesses, the success rate of our algorithm will increase along with the size of the explored search space.

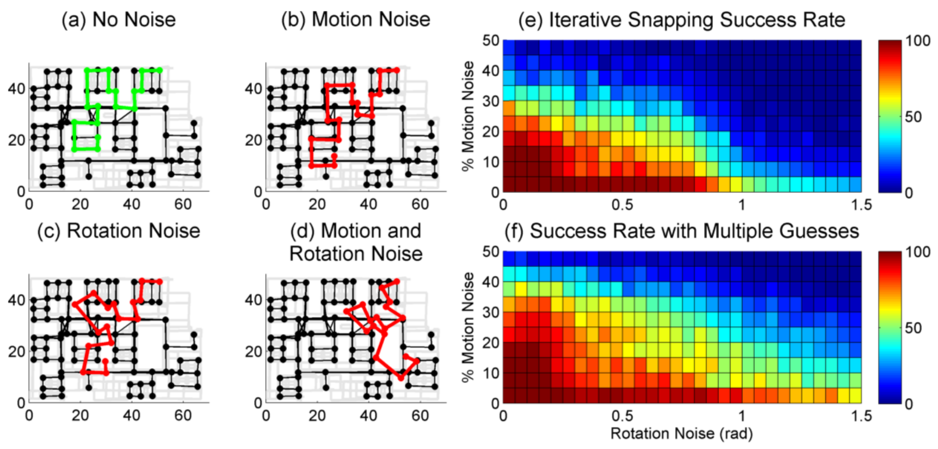

Fig. 4. Testing robustness of our snapping algorithm by injecting noise into a (a) ground truth path. (b) Motion only noise $\eta = \pm 25\%$. (c) Rotation only noise $\epsilon = \pm .8rad$. (d) Both motion and rotation noise $\eta = \pm 25\%, \epsilon = \pm .8rad$. (e) Success rate of snapping injected with various combinations of noise. (f) Improved success when the iterative algorithm is augmented by maintaning a set of 5 unique guesses.

5.2 Success with Real-World Data

We evaluate odometry from wheel encoders of an omni-directional robot collected from the path traversed in Figure 1 that travels almost 250 m. This environment is not nearly as challenging as the simulated environment because it requires fewer difficult decisions. In Figure 5.2, we show how our snapping algorithm iteratively corrects the error in odometry. Each decision is a simple comparison between the trajectory segment and its corresponding hallways because the prior trajectory segments have already been snapped. The noise model discovered reveals that corrections to the wheel encoder odometry resulted in motion noise within 20% and rotation noise within .5 radians. These coarse noise estimates are not as accurate when compared to the ground truth in Figure 1 because they are influenced by the marked positions of the hallway intersections. Nevertheless, these coarse estimates can help to reveal patterns in the normal behavior of snapping and outliers could help to suggest when incorrect snapping decisions are made.

6 CONCLUSION

Our snapping algorithm can recover paths with similar shapes to the trajectories captured by odometry. Forcing the trajectory to fit the marked hallways allows us to focus on important decisions at intersections to ensure global robustness. In addition, it allows us to recover coarse noise estimates from the modifications required to fit the trajectory to the hallways. Many opportunities remain to

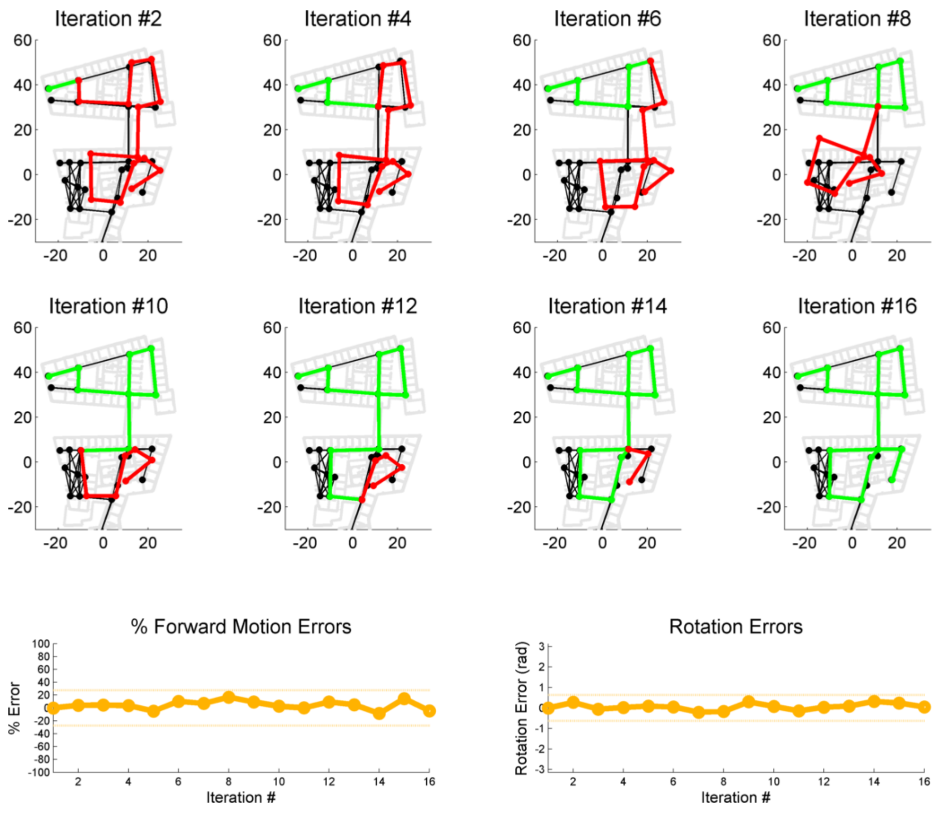

Fig. 5. Snapshot of several iterations of snapping from trajectory segments extracted from real wheel encoder odometry. The snapped segments (green) and unsnapped segments (red). Snapping results in modifications to the trajectory segments and these can be perceived as coarse noise estimates that are shown at the bottom.

consider curved hallways, combining odometry with other sensors, and recovery from incorrect snapping decisions.

References

1. Johann Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *Robotics and Automation, IEEE Transactions on*, 12(6):869–880, 1996.
2. 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.
3. Joseph Huang, David Millman, Morgan Quigley, David Stavens, Sebastian Thrun, and Alok Aggarwal. Efficient, generalized indoor wifi graphslam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1038–1043. IEEE, 2011.
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. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.