Cost-based Registration using A Priori Data for Mobile Robot Localization

Ling Xu Anthony Stentz

CMU-RI-TR-08-05

January 2008

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

A major challenge facing outdoor navigation is the localization of a mobile robot as it traverses a particular terrain. Inaccuracies in dead-reckoning and the loss of global positioning information (GPS) often lead to unacceptable uncertainty in vehicle position. We propose a localization algorithm that utilizes cost-based registration and particle filtering techniques to localize a robot in the absence of GPS. We use vehicle sensor data to provide terrain information similar to that stored in an overhead satellite map. This raw sensor data is converted to mobility costs to normalize for perspective disparities and then matched against overhead cost maps. Cost-based registration is particularly suited for localization in the navigation domain because these normalized costs are directly used for path selection. To improve the robustness of the algorithm, we use particle filtering to handle multi-modal distributions. Results of our algorithm applied to real field data from a mobile robot show higher localization certainty compared to that of dead-reckoning alone.

Contents

1 Introduction

Many autonomous ground vehicles operating outdoors use an aerial map to navigate from one point to the next. In order to ensure little deviation from this path, ground vehicles need to know their exact location during their traversal. The most common way to determine this position is from the Global Positioning System (GPS). However, GPS is sometimes unavailable due to occlusion, technical failures or jamming. One contingency to GPS is dead-reckoning (i.e., inertial sensing plus odometry) a method prone to increasing error over distance and time. In these cases, prior knowledge of the environment can be very useful in correcting odometry errors to more accurately locate the robot.

Prior knowledge is largely available in the form of overhead maps such as satellite imagery for different regions of the world and is reasonably easy to obtain. While the resolution of these maps may be coarse, the maps typically give a comprehensive view of an area. Additionally, the vehicle usually possesses a perception system that can sense the terrain around the robot to produce a more detailed local map. A natural method for registration in localization algorithms compares the global and local maps to find data matches at particular positions.

Current localization techniques typically employ visual landmarks as markers to align two sets of data [1] [2]. Literature exists that cite different methods of finding and extracting distinguishing features to be landmarks such as [3] [4]. However, the data from the overhead imagery and local sensors may differ greatly due to perspective differences. For instance, tree trunks are visible when viewed from the local perspective while only the canopies are visible from above making individual feature comparisons difficult. These data inconsistencies drastically complicate feature tracking in outdoor environments. In constrast, we use mobility costs. Mobility costs are derived from terrain features and specify the difficulty of traversing a particular patch of terrain. Thus, mobility cost maps minimize reliance on a single landmark for localization and help reduce perspective effects by capturing the quality of the environment rather than its features or characteristics.

In order to localize based on cost map matches, we use a particle filter to manage estimates of the robot's state. Current methods for state estimation include Kalman filters and particle filters [5]. Recent localization algorithms have employed these two techniques for position estimation such as [6] [7]. Kalman filters assume the belief state is unimodal, typically a Gaussian distribution, which may not accurately reflect the particular state distribution being modelled [8]. Alternatively, a particle filter, which is a sample-based estimation Monte Carlo technique, can handle multi-modal distributions by maintaining competing hypotheses for the robot position [9]. Finally, particle filters are generally more space efficient because they sample a space instead of modelling the entire space.

Our contribution consists of applying a well-established estimation method to overhead and ground mobility cost maps to localize an outdoor mobile robot. While particle filters and cost maps have each been used for localization, the combination of the two has not been tested. We show that despite cost discrepancies, a particle filter combined with a simple similarity metric demonstrates promising results for localization. We compare our method to pure dead-reckoning on a test course where the robot traveled

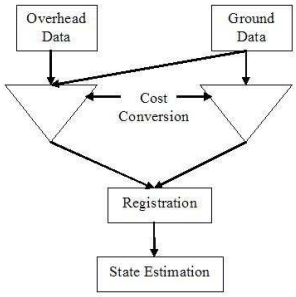

Figure 1: Illustration of our approach

over 3000m.

This paper is organized in a top-down manner. We present an overview of the approach in Section II. We show more details and examples of the different algorithm components in Section III. We illustrate and analyze results conducted on data collected from field tests in Section IV. Finally we present our conclusions in Section V.

2 Approach

Our approach localizes a mobile robot using cost-based registration to estimate the robot's position. Our approach process is illustrated in Figure 1. We used commercially available satellite imagery with a 30cm resolution as the prior data. The local data was gathered using the perception system on the robot. To factor out perspective differences, we transformed the two data sets into mobility cost maps. During each cycle of a particle filter, we register the two maps to produce a state estimate for the vehicle. While the conversion to cost maps is not the focus of this paper, we explain the conversion process to illustrate the registration process.

We process raw image and range data from terrain features such as vegetation, ground, and rocks into a single mobility value [10]. To support a grid-based representation, the maps are divided into grid cells and this conversion from raw features to a combined mobility cost occurs for each grid cell in both the overhead and local maps. The overhead map costs are made consistent with the ground costs using self-supervised learning [11]. However, this process cannot remove all the inconsistencies between the two cost maps generated by occlusion (the obstruction of certain features by other features) and aliasing (the mapping of multiple features to a single object). Aliasing effects result in multi-modal position matches where one item in the local map matches more than one item in the global map. We used a particle filter because of its ability to handle the multi-modal nature of the map matches.

For position estimation, the particle filter keeps a sample of particles which denote

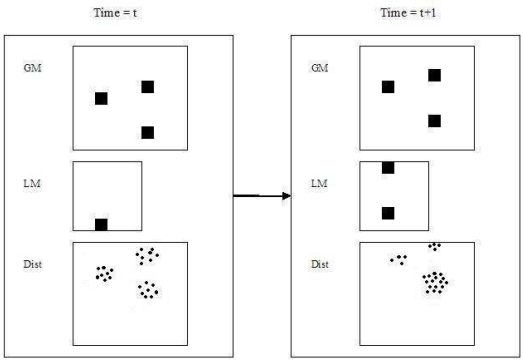

Figure 2: This example captures the capability of the particle filter in maintaining multiple hypotheses. At time step t, the global map (GM) contains three features that match the local map (LM) resulting in the distribution of the particles shown in the bottom figure. We assume the vehicle is centered in the local map. After it moves at time step $t + 1$, the vehicle collects more information of the scene, updates the local map (LM), and the particle distribution shifts to favor the bottom right location.

(x, y) vehicle positions. At each cycle, the algorithm operates in two main steps: prediction and correction. The prediction step estimates the motion of the particles using the vehicle's motion model. We used dead-reckoning as our motion model defined as commanded vehicle movement plus Gaussian noise. With pure dead-reckoning, the position error of the vehicle grows indefinitely within the Gaussian noise bounds.

The correction step decreases the effects of the motion noise by adjusting the particle weights using registration. The algorithm calculates the quality of matches using a similarity measure. Centering the local cost map at each particle position, the similarity metric returns the value of the match at the specific location. This value determines the removal of certain particles and the retainment of others. In this manner, the particle filter retains the vehicle positions with the most map matches. Figure 2 shows an example where the particle filter first finds three matches between the global and local maps at time step t and then reduces these three hypotheses down to one when the vehicle captures more environmental information at time step $t + 1$. This technique also helps minimize the effects of sporadic false matches because it considers the weighted average of the samples at each comparison cycle.

Since the mobility cost map is also used for path generation, mismatches using these maps are more significant than mismatches using another map format.

3 Algorithm Details

This section presents the details of our approach (Algorithm 1). In the algorithm, (a_t^i, w_t^i) represents a particle, in particular particle i at time t where a_t^i represents the (x, y) location and w_t^i represents the corresponding weight. We initialize the particle filter with M particles of equal probability at the correct vehicle location (Alg. 1 line 1). In the prediction step, the samples are projected forward in time using the motion model (Alg. 1 line 3). In the correction step, the local mobility cost map s_t is matched to the overhead cost map at each particle pose and a similarity metric assigns a probability $p(s_t|a_t^i)$ to the match. Then the algorithm uses the registration probability to update weights of each particle (Alg. 1 line 4). After the prediction and correction steps, the algorithm returns the estimated position (Alg. 1 line 5-6). Finally in the resampling step, if the effective sample size drops below a certain threshold, a new set of particles is selected from the current distribution (Alg. 1 line 7-9). α is the resampling factor which determines this resampling threshold. Next, we will unpack each of the four major steps individually.

Algorithm 1 Particle Filter

1: $(a_0^i, w_0^i) \Leftarrow (a_0^i, 1/M)$
2: **for** t=1 to T **do**
3: $a_t^i \Leftarrow a_{t-1}^i + \delta a_{t-1}^i + N(0, \sigma^2)$ {Prediction Step}
4: $w_t^i \Leftarrow w_t^i \cdot p(s_t|a_t^i)$ {Correction Step}
5: $a_t^{est} \Leftarrow \frac{\Sigma_i a_t^i \cdot w_t^i}{\Sigma_i w_t^i}$ {Position Estimation}
6: return a_t^{est}
7: **if** ESS $< \alpha$M **then**
8: Resample (a_t^i, w_t^i) by sampling from weights {Resampling}
9: **end if**
10: **end for**

3.0.1 Prediction Step

In the prediction step, the filter estimates the motion of the particles at every cycle using a motion model, in this case the dead-reckoning of the vehicle. In order to accurately simulate dead-reckoning, we add noise to the motion model to model odometry errors. The noise is assumed to be a Normal distribution with the spread of the distribution proportional to the distance traveled. We assume the noise distribution to have a spread of 3 standard deviations from the mean that is equal to 5% of the total distance traveled. For the motion noise in our experiments, we chose the standard deviation (σ) to be equal to the amount the vehicle moves every cycle. We also assume that the noise samples are independent with respect to time. Hence the predicted motion becomes the distance traveled δa_{t-1}^i plus a small amount of Gaussian noise $\mathcal{N}(0, {\delta a_{t-1}^i}^2)$ (Algorithm 2). The predicted motion is independently calculated for each axis component of the displacement.

For example, imagine the robot starts out at $(0, 0)$ and commands a movement of

Algorithm 2 Motion Update for time t

1: **for** i=1 to M **do**
2: $\quad a_t^i \Leftarrow a_{t-1}^i + \delta a_{t-1}^i + \mathcal{N}(0, \delta {a_{t-1}^i}^2)$
3: **end for**

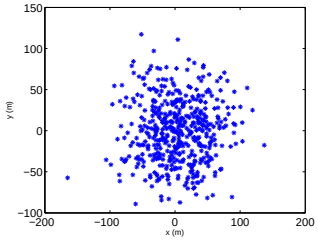

Figure 3: Particle locations using only the motion model after the vehicle has traveled roughly 3 km

$0.5m$ in the x direction and $0.7m$ in the y direction. Assume that all the particles are at $(0, 0)$. The motion model will predict that the particles moved to $(0.5 + \alpha_i, 0.7 + \beta_i)$ where α_i is randomly drawn from $\mathcal{N}(0, 0.5^2)$ and β_i is randomly drawn from $\mathcal{N}(0, 0.7^2)$.

Fig. 3 shows the location of the set of particles after a test run using the motion model alone. After 8000 iterations, the vehicle has traveled roughly 3 km. At the end of the run, the particles represents a Normal distribution with a σ of 56m, roughly 1.7% of the total travel distance. Hence, the motion model accurately estimates the dead-reckoning error accumulated through the test run since 3σ is approximately 5% of 3 km.

3.0.2 Correction Step

In the correction step, the particle filter corrects the prediction using a Bayesian update (Algorithm 3). Every cycle, new sensor data produces a new cost map. The algorithm centers the local cost map S at every particle position on the global cost map A, and calculates the quality of the match using the sum of squared difference metric between corresponding map cells $SSE_{a_t^i}$ (Alg. 3 line 2). This measurement is then converted into a likelihood probability $p(s_t|a_t^i)$ by taking the fraction of the difference value over the maximum difference value $maxSSE$ (Alg. 3 line 5). Using this similarity metric, the larger the likelihood probability, the better the match. An example of a error distribution generated from real data is shown in Fig. 4. In this figure, there is a clear match around (-0.4m, 1.6m) indicating the most likely vehicle position.

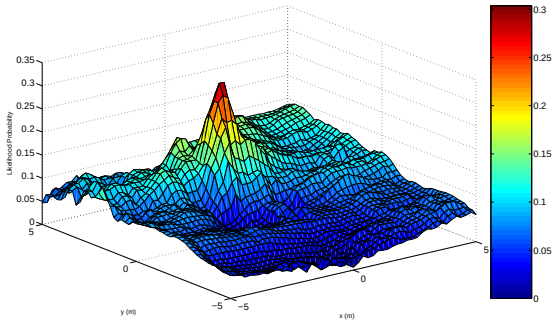

Figure 4: Matching positions between global and local cost maps. Position $(-0.4m, 1.6m)$ is the best match using the sum of squared differences metric.

Algorithm 3 Sensor Update for time t

1: **for** i=1 to M **do**
2: $SSE_{a_t^i} \Leftarrow \Sigma_{(x,y)}(A(x,y) - S(x,y))^2$
3: **end for**
4: $maxSSE \Leftarrow \underline{max}(SSE)$
5: $p(s_t|a_t^i) \Leftarrow \frac{maxSSE - SSE_{a_t^i}}{maxSSE}$

3.0.3 Registered Position

The estimated pose or the most likely robot location is the weighted average of the particles (1).

$$a_t^{est} = \frac{\sum_i w_t^i \cdot a_t^i}{\sum_i w_t^i} \tag{1}$$

3.0.4 Resampling Step

In the resampling step, the algorithm samples the particles based on their weights to ensure no particle has an extremely small weight. Resampling will generate more particles (a_t^i, w_t^i) if w_t^i is large and generate fewer particles if w_t^i is small.

In order to determine when to resample, we employ the technique of rejection control [12]. At every iteration, the number of particles with weight close to zero is calculated using a heuristic called the effective sample size:

$$ESS_t = \frac{m}{1 + cv^2{}_t} \tag{2}$$

The effective sample size, which describes the number of independent samples, is calculated from the coefficent of variation of the particles:

$$cv^2{}_t = \frac{var(w_t^i)}{\mathbb{E}^2(w_t^i)} \tag{3}$$

During tests, we use a resampling factor (α) of 0.1 to determine when resampling should occur. If the effective sample size is higher than the predetermined threshold, then the particles and their weights are maintained through the next iteration (Fig. 5). If the effective sample size is lower than the checkpoint threshold, resampling is performed. For resampling, we use the select-with-replacement algorithm that randomly samples the original particles based on their weights resulting in a subset of particles with the highest likelihood values (Fig. 6). Finally the weights are normalized to be equal for all the particles. In tests, we chose the subset size to be 10% of the original set size.

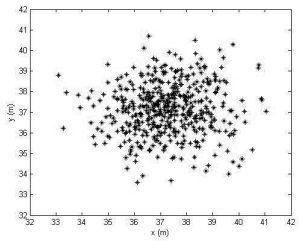

Figure 5: Before resampling: Sensor update increases the weights of certain particles while lowering the weights of others

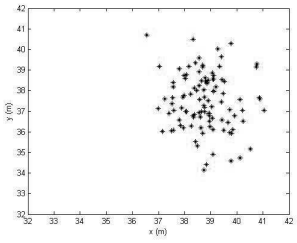

Figure 6: After resampling: Resampling retains the particles with higher weights and removes particles with lower weights

Figure 7: Crusher hybrid electric vehicle used in field tests

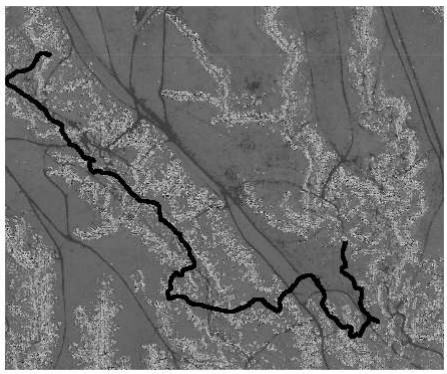

Figure 8: A priori cost map at 30cm resolution with a size of 974m by 806m. The black line represents the vehicle's path through the terrain.

4 Results

The robot used for field testing was a large, six-wheeled vehicle equipped with laser rangefinders and cameras for on-board perception (Fig. 7). To test the algorithm, we used data collected from field tests conducted on off-road terrain containing an assortment of vegetation, slopes, ditches, trees, and rocks. This data also included the true vehicle location collected using GPS. A path planner used the cost maps generated from overhead and local data to find the lowest cost route through the terrain.

The map cost values range from 0 to 65535 with 0 being unknown and 65535 being completely blocked. Lower cost indicate more traversable map cells. Because the on-board sensors and overhead mapping process differ significantly in data resolution and occlusion properties, the extreme cost values (i.e., 0 and 65535) are discarded to facilitate better matching.

Fig. 8 shows the area that the robot traversed. It contains both open areas with few terrain features and dense areas with many more features. We tested the localization algorithm in simulation on this area by ignoring the GPS information. We ran the experiment on the tree-covered portion of the terrain, since a vehicle is most likely to lose GPS in these areas; however, these areas are also likely to have unique cost features for matching.

During testing, the robot generated local maps with a 20cm resolution and a size of 24m by 24m. The robot was centered on each of the maps. The global map had a 30cm resolution and was 974m by 806m in size. We initialized the algorithm with 484 particles at the correct starting pose. We assumed that before the robot traveled into the test region, GPS was available and position was known. Once the vehicle entered the region, GPS became unavailable. In the map, this region starts at the upper left hand

side and continues down through the more feature-rich terrain, depicted by the black path. The localization algorithm was invoked once per meter of robot travel. During the entire test run, the robot traveled approximately 2.263 km in the x direction and 2.072 km in the y direction giving a total distance of 3.068 km.

Fig. 9 shows the Euclidean distance between the estimated pose (producted from simulated experiments) and the correct pose (collected from the dataset) at each frame. While some frames benefit from registration, for other frames, localization errors are high. The estimated position deviates as little as 1m and as much as 45m from the true position of the robot (Fig. 10). One reason is due to ambigious matches in the registration process particularly when the likelihood probability distribution appears to be multi-modal. These ambiguities or aliasing effects lead to several high scoring matches between the sensor and overhead data as opposed to just one. In frames containing the aliasing effects, the algorithm fails to choose the correct location. An instance of an aliasing effect occurs around frame 6000 where the scene is very sparse with few features against which to register. The graph of the position matches indicate several mismatches (Fig. 11). The particle filter could not recover from the mismatches because the vehicle stayed in the open area for almost 200 cycles before reaching a region with more distinct features. One way to solve this problem is to detect when the distribution is multi-modal, select the best position using a heuristic, and renormalize the particles based on the position selected. This method will mitigate aliasing effects by removing the ambiguities from the data.

Localization errors also occur because of perspective effects and occlusion that are not completely removed by the cost conversion. For instance, the global and local maps at frame 4700 are shown in Fig. 13. While a high cost feature appears in the overhead map (circled part), only a portion of the feature appear in the sensor map. The remainder of the feature is occluded making it difficult to match the data exactly due to the missing part. This results in the registration returning an incorrect match. Whereas the global and local data should match at the center of the frame, the best match using sum of squared difference at positions within a 10m by 10m square around the true pose is at position $(5m, -4m)$, 6.7 meters off (Fig. 12). However the particle filter recovers from this mismatch because the large weights for the correct position offset the single mismatch. One way of further removing occlusion effects during the cost conversion process is to only use cost features that can be measured easily in both overhead and sensor data. For instance, features such as trees would be disregarded due to potentially large perspective differences.

Fig. 14 shows the position distributions between dead-reckoning plus registration and dead-reckoning alone at the end of the test. While the dead-reckoning distribution contains no bias, the standard deviation of the distribution is fairly large at 56.7m. On the other hand, the standard deviation of the localization distribution is much smaller, at 4.8m. The two standard deviations indicate that the uncertainty using only dead-reckoning is greater than the uncertainty using dead-reckoning plus registration. In other words, dead-reckoning errors can grow up to 156m due to Gaussian noise. However, incorporating registration pushes the estimated vehicle position to be 35m away from the correct position. This result arises because of the high number of map mismatches that occur in the second half of the run. In this comparison, dead-reckoning possesses zero bias because it is produced from simulation while the registration re-

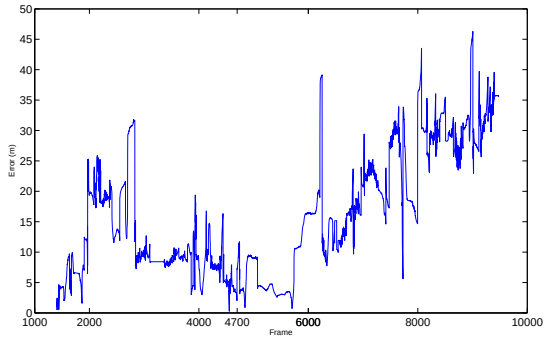

Figure 9: Registration error when compared to GPS ground truth at every frame

sults were obtained through experiments and therefore subject to bias due to position mismatches.

One way to interpret the results is to use risk analysis to find the amount of risk involved in each method. We consider risk as a weighted mean of a particular loss function:

$$risk_t = \Sigma_i L(a_t^i) \cdot w_t^i \tag{4}$$

For instance, we analyzed the results using a popular loss function, the quadratic function $L(x) = (a_t^i - a_t^{est})^2$. We chose this because we wanted to penalize more for uncertainty in the estimate. Using the particles and their weights at the completion of the test run, the risk calculated for the localization results is 31693.5 while the risk for dead-reckoning is 80688.01. Thus using this particular loss function, pure dead-reckoning is more than twice as risky as cost-based registration.

5 Conclusion

When GPS is absent, relying on dead-reckoning alone can be risky because the robot can become lost in an environment due to error accumulation. Incorporating environmental information to help track the position of a vehicle can be very useful in robot localization. In this paper, we proposed a particle filter approach that uses mobility cost maps to address the position tracking problem. However two main obstacles to perfect cost-based registration are aliasing and occlusion effects. Despite these issues, sample-based localization produces a position distribution with a much smaller loss when compared to the distribution of pure dead-reckoning. Depending on the loss function used to analyze the risk of each approach, localization via registration may perform better. Therefore, cost-based registration for robot localization is a promising method for position tracking.

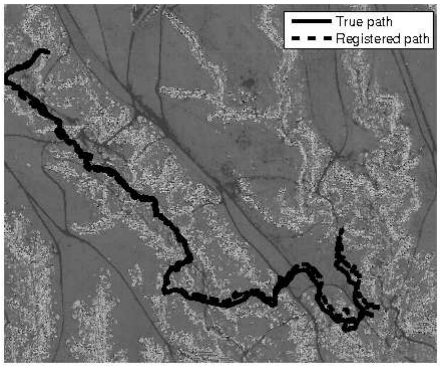

Figure 10: Comparison between the true path (solid line) and the registered path (dashed line)

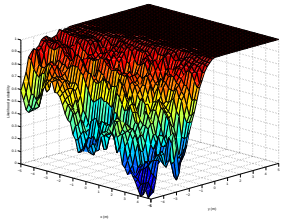

Figure 11: Mismatch caused by aliasing (Higher cost values represent better matches) The correct match is at the center of the graph

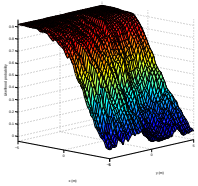

Figure 12: Mismatch caused by occlusion (Higher cost values represent better matches) The correct match is at the center of the graph

Acknowledgments

This work was sponsored by the National Science Foundation under the graduate fellowship program. The data used in this work was produced under sponsorship of the Defense Advanced Research Projects Agency (DARPA) via contract ”Unmanned Ground Combat Vehicle - PerceptOR Integration” (contract number MDA972-01-9-005). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The authors thank David Silver and Boris Sofman for their assistance with the data.

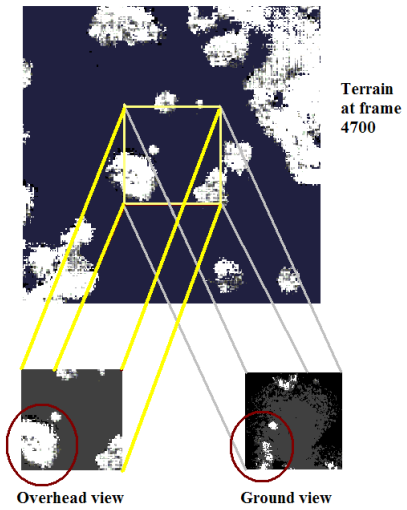

Figure 13: The bottom left cost map gives the overhead view of the area at frame 4700. The bottom right cost map gives a ground view of the same area. The two circles indicate a feature that shows up fully in the overhead view, but is occluded in the ground view. Lighter colors indicate higher cost.

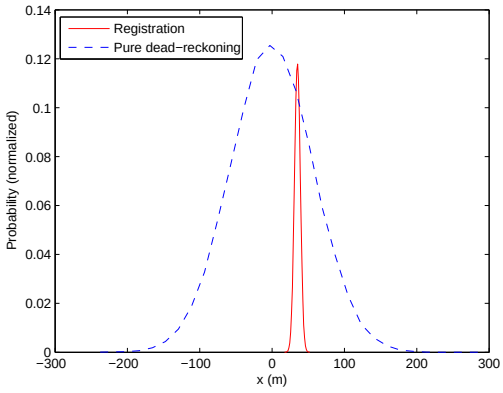

Figure 14: Position distributions of particles resulting from registration and dead-reckoning

References

[1] M. Betke and K. Gurvits. Mobile robot localization using landmarks, 1994.

[2] Sebastian Thrun. Bayesian landmark learning for mobile robot localization. Machine Learning, 33(1):41–76, 1998.

[3] S. Thrun. Finding landmarks for mobile robot navigation. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1998.

[4] M. Adams, Sen Zhang, and Lihua Xie. Particle filter based outdoor robot localization using natural features extracted from laser scanners. In ICRA '04. 2004 IEEE International Conference on Robotics and Automation Volume II, pages 1493–1498, 2004.

[5] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. IEEE Pervasive Computing, 2003.

[6] Dieter Fox, Sebastian Thrun, Wolfram Burgard, and Frank Dellaert. Particle filters for mobile robot localization. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, Sequential Monte Carlo Methods in Practice, New York, 2001. Springer.

[7] Hyun Myung, Hyoung-Ki Lee, Kiwan Choi, Seok-Won Bang, Yong-Beom Lee, and Sang-Ryoung Kim. Constrained kalman filter for mobile robot localization with gyroscope. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 442–447, 2006.

[8] Concept Derivation And. Kalman and extended kalman filters:.

[9] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In IEEE International Conference on Robotics and Automation (ICRA99), May 1999.

[10] David Silver, Boris Sofman, Nicolas Vandapel, James Bagnell, and Anthony (Tony) Stentz. Experimental analysis of overhead data processing to support long range navigation. In IEEE International Conference on Intelligent Robots and Systems (IROS), October 2006.

[11] Boris Sofman, Ellie Lin, James Bagnell, Nicolas Vandapel, and Anthony (Tony) Stentz. Improving robot navigation through self-supervised online learning. In Proceedings of Robotics: Science and Systems, August 2006.

[12] J. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling and resampling, 2000.